625-EMD-011

EOSDIS Maintenance and Development Project

Training Material for the EMD Project
Volume 11: Database Administration

July 2004

Raytheon Company
Upper Marlboro, Maryland

Training Material for the EMD Project
Volume 11: Database Administration

July 2004

Prepared Under Contract NAS5-03098
CDRL Item 23

RESPONSIBLE AUTHOR

Ralph E. Fuller /s/ 7/13/2004

Ralph E. Fuller Date
EOSDIS Maintenance and Development Project

RESPONSIBLE OFFICE

Mary S. Armstrong /s/ 7/13/2004

Mary Armstrong, Deputy Program Manager Date
EOSIDS Maintenance and Development Project

Raytheon Company
Upper Marlboro, Maryland

625-EMD-011

This page intentionally left blank.

Preface

This document is a formal contract deliverable. It requires Government review and approval within 45 business days. Changes to this document will be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The EMD Project Office
Raytheon Company
1616 McCormick Drive
Upper Marlboro, Maryland 20774-5301

Revision History

Document Number	Status/Issue	Publication Date	CCR Number
625-EMD-011	Original	July 2004	04-0384

This page intentionally left blank.

Abstract

This is Volume 11 of a series of lessons containing the training material for the Earth Observing System Data and Information System (EOSDIS) Maintenance and Development (EMD) Project. This lesson provides a detailed description of the process required to perform the tasks associated with database administration.

***Keywords**:* training, database administration, course objective, metadata, Release 7.

This page intentionally left blank.

Contents

ECS Overview

ECS Database Administrator (DBA) Responsibilities

Starting and Stopping Servers

Creating Database Devices and Logical Volumes

Installing Databases and Patches

Configuring Databases

Working with Indexes, Segments, and Caches

Backing Up and Recovering Data

Establishing Database Security

Copying, Replicating, and Extracting Data

Replication System Administration

Performance Monitoring, Tuning, and Problem Reporting

Ensuring Database Quality

Sybase Troubleshooting

Oracle Procedures

Practical Exercises

Slide Presentation

This page intentionally left blank.

Introduction

Identification

Training Material Volume 11 is part of Contract Data Requirements List (CDRL) Item 23, which is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Maintenance and Development (EMD) Contract (NAS5-03098).

Scope

Training Material Volume 11 describes the process and procedures for database administration. This lesson is designed to provide the operations staff with sufficient knowledge and information to satisfy all lesson objectives.

Purpose

The purpose of this Student Guide is to provide a detailed course of instruction that forms the basis for understanding data distribution. Lesson objectives are developed and will be used to guide the flow of instruction for this lesson. The lesson objectives will serve as the basis for verifying that all lesson topics are contained within this Student Guide and slide presentation material.

Status and Schedule

This lesson module provides detailed information about training for the current baseline of the system. Revisions are submitted as needed.

Organization

This document is organized as follows:

Introduction:	The Introduction presents the document identification, scope, purpose, and organization.
Related Documentation:	Related Documentation identifies parent, applicable and information documents associated with this document.
Student Guide:	The Student Guide identifies the core elements of this lesson. All Lesson Objectives and associated topics are included.
Slide Presentation:	Slide Presentation is reserved for all slides used by the instructor during the presentation of this lesson.

This page intentionally left blank.

Related Documentation

Parent Documents

The parent documents are the documents from which the EMD Training Material's scope and content are derived.

423-41-01	Goddard Space Flight Center, EOSDIS Core System (ECS) Statement of Work
423-46-03	EMD Task 101 Statement of Work For ECS SDPS Maintenance
423-46-02	Contract Data Requirements Document for EMD Task 101 ECS SDPS Maintenance

Applicable Documents

The following documents are referenced within this EMD Training Material, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document:

420-05-03	Goddard Space Flight Center, Earth Observing System (EOS) Performance Assurance Requirements for the EOSDIS Core System (ECS)
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) (ECS F&PRS)
423-46-01	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) Science Data Processing System (EMD F&PRS)

Information Documents

Information Documents Referenced

The following documents are referenced herein and amplify or clarify the information presented in this document. These documents are not binding on the content of the EMD Training Material.

420-EMD-001	Release 7 Implementation Earth Science Data Model for the EMD Project
609-EMD-001	Release 7 Operations Tools Manual for the EMD Project
611-EMD-001	Mission Operation Procedures for the EMD Project

910-TDA-021	SYBASE SQLServer 11.0.x ALL DAAC Database Configurations
910-TDA-022	Custom Code Configuration Parameters for ECS
920-TD*x*-001	Hardware-Design Diagram
920-TD*x*-002	Hardware-Software Map
920-TD*x*-009	DAAC HW Database Mapping
FB9401V2	EOSDIS Core System Science Information Architecture

Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of the EMD Training Material.

305-EMD-001	Release 7 Segment/Design Specification for the EMD Project
311-EMD-001	Release 7 Data Management Subsystem (DMS) Database Design and Database Schema Specifications for the EMD Project
311-EMD-002	Release 7 INGEST (INS) Database Design and Schema Specifications for the EMD Project
311-EMD-003	Release 7 Planning and Data Processing Subsystem Database Design and Schema Specifications for the EMD Project
311-EMD-004	Release 7 Science Data Server Database Design and Schema Specifications for the EMD Project
311-EMD-005	Release 7 Storage Management and Data Distribution Subsystems Database Design and Database Schema Specifications for the EMD Project
311-EMD-006	Release 7 Subscription Server Database Design and Schema Specifications for the EMD Project
311-EMD-007	Release 7 Systems Management Subsystem Database Design and Schema Specifications for the EMD Project
311-EMD-008	Release 7 Registry Database Design and Schema Specifications for the EMD Project
311-EMD-009	Release 7 Product Distribution Subsystem (PDS) Database Design and Database Schema Specifications for the EMD Project
311-EMD-010	Release 7 NameServer Database Design and Schema Specifications for the EMD Project
311-EMD-011	Release 7 Order Manager Server Database Design and Schema Specifications for the EMD Project

311-EMD-012	Release 7 Spatial Subscription Server Database Design and Schema Specifications for the EMD Project
311-EMD-013	Release 7 Data Pool Database Design and Schema Specifications for the EMD Project
313-EMD-001	Release 7 ECS Internal Interface Control Document for the EMD Project
152-TP-001	ACRONYMS for the EOSDIS Core System (ECS) Project
152-TP-003	Glossary of Terms for the EOSDIS Core System (ECS) Project

This page intentionally left blank.

Database Administration

Lesson Overview

This lesson will provide you with the tools needed to perform the various tasks required to administer and maintain the database and structure management for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) during maintenance and operations.

Lesson Objectives

Overall Objective - This lesson provides a detailed description of the different tasks required to maintain the database and structure management for ECS, provide the operations interface to perform database administration utilities such as product installation and disk storage management, managing user accounts and privileges, backup and recovery, monitoring physical allocation of database resource information, loading metadata and maintaining metadata.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures required to perform database administration.

Specific Objective 1 - The student will create new database devices, allocate appropriate disk space to house the new database, and maintain database segments including managing and monitoring the use of available disk space, memory, connection error logs, state of transaction logs, device problems, etc.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of new database devices, the allocation of appropriate disk space, and maintenance of database segments.

Specific Objective 2 - The student will start and shutdown the SQL server.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating the startup and shutdown of the SQL server.

Specific Objective 3 - The student will perform database user account and access privilege procedures including:

- Creating user accounts.
- Granting and revoking access privileges for data retrieval, insertion, deletion and update of objects.
- Granting and revoking roles for SQL server users groups.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of user accounts and the granting and revoking of access privileges.

Specific Objective 4 - The student will perform database security and auditing procedures.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to database security and auditing procedures.

Specific Objective 5 - The student will perform database integrity monitoring.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to database integrity monitoring.

Specific Objective 6 - The student will perform database backups on a regular or on-demand basis.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to database backups.

Specific Objective 7 - The student will perform a recovery of the database following a system failure or on-demand.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to database recovery.

Specific Objective 8 - The student will configure databases unique to ECS DAACs including:

- Making database size estimates and planning.
- Preparing database-unique attributes.
- Preparing database reports.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to database configuration specific to the ECS DAACs.

Specific Objective 9 - The student will perform database tuning and performance monitoring procedures including:

- Design and indexing.
- Responding to queries.
- Monitoring and boosting performance.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will perform without error the procedures relating to database tuning and performance monitoring.

Specific Objective 10 - The student will describe Sybase Replication Server Administration.

Condition - The student will be given a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*, 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a functioning system.

Standard - The student will describe Sybase Replication Server Administration.

Importance

ECS relies on vast amounts of data from the science user perspective and from a maintenance and operations perspective. Accurate information stored in the science databases allows science users to access necessary data quickly. Similarly, databases that maintain operational data must be kept current in order for routine and specialized administrative tasks to be performed.

Special Instructions on Procedures

All procedures in this training guide assume that you are logged in to a system server or workstation as yourself.

Many database administration tasks are performed by using scripts that are run from the command line. Some scripts may require modification that will require you to use a text editor of your choice. These procedures do not give keystroke-by-keystroke information on how to edit the file; they only instruct you to make the changes required to perform the task.

ECS Overview

The EOSDIS Core System (ECS)

The EOSDIS Core System (ECS) is a large-scale data and information system. A key to understanding ECS, therefore, is understanding ECS databases. Similarly, the first step in understanding ECS databases is understanding ECS design.

General Design

ECS is designed to:

- Receive data from external sources
- Save those data in either long-term or permanent storage
- Produce higher-level data products from the received data
- Support access to the data by scientists and other registered clients

Information Model

The ECS information model characterizes earth science data as a data pyramid consisting of broad, multi-layered data categories as shown in Figure 1. Logical collections of data, based on their expected relationships, are developed to capture the variability in remote sensing instruments, science disciplines, and other characteristics of the earth science community. For example, some products have related properties (e.g., cloud type and cloud drop size) while other products are dissimilar (e.g., land vegetation indices and ocean productivity), which suggest certain logical groupings. Characteristics are often similar across a particular science discipline and across products generated from a given instrument but different among the various provider sites because of differing science disciplines focus and organizational autonomy.

Metadata. *Metadata are data about data that are provided to ECS by the external data provider or the generating algorithm.* They describe characteristics of data origin, content, format, quality, and condition. They also provide information to process and interpret data. Metadata are required for access all data in the ECS. The *Role of Metadata in EOSDIS* (160-TP-013-001) provides a good introduction to how metadata are used in ECS.

An earth science metadata model supports the data standardization necessary for total system interoperability within a heterogeneous, open systems environment. (Refer to the latest version of the *Earth Science Data Model*; e.g., 420-EMD-001, Release 7 Implementation Earth Science Data Model for the EMD Project.) The ECS data model includes diagrams that illustrate the relationships of classes, the attributes contained within the classes, the characteristics of the relationships between classes, and the attribute specifications.

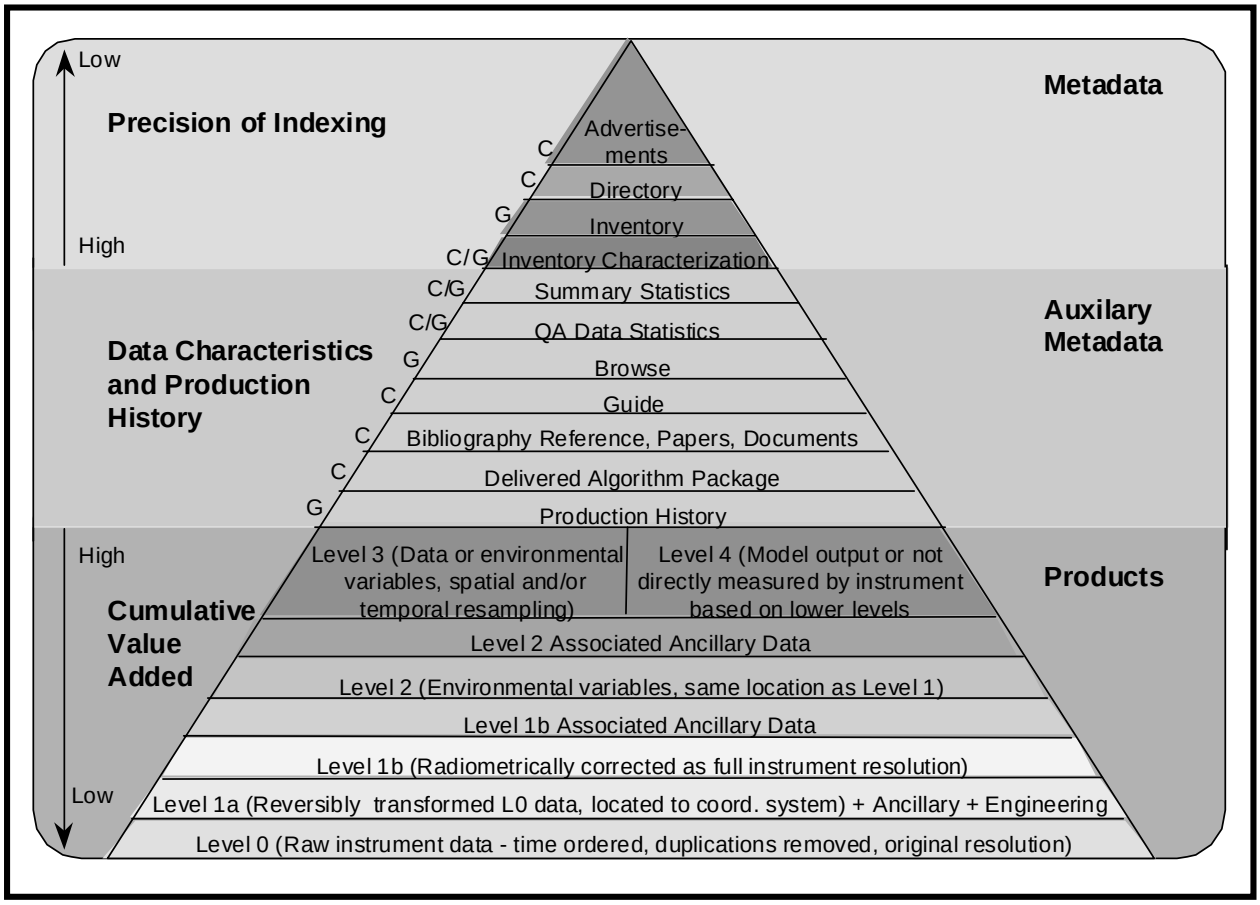

Figure 1. ECS Information Model

Attributes are descriptors of data populating searchable database fields, enabling finite classification of data residing in ECS. Attributes can either be collection-level or granule-level attributes and either core or product-specific attributes. A collection is a grouping of related science data. A granule is the smallest aggregation of data that is independently managed (i.e., ingested, processed, stored, or retrieved) by the ECS. The majority of attributes in the data model are collection-level attributes, which means that they apply to all granules in the collection.

Data Products. *Data products are a processed collection of one or more parameters packaged with associated ancillary and labeling data and formatted with uniform temporal and spatial resolution, e.g., the collection of data distributed by a data center or subsetted by a data center for distribution.* There are two types of data products:

- Standard, which is a data product produced at a DAAC by a community consensus algorithm for a wide community of users.
- Special, which is a data product produced at a science computing facility by a research algorithm for later migration to a community consensus algorithm and can be archived and distributed by a DAAC.

Data products are categorized by levels, which are described in shown in Figure 2 and described in Table 1.

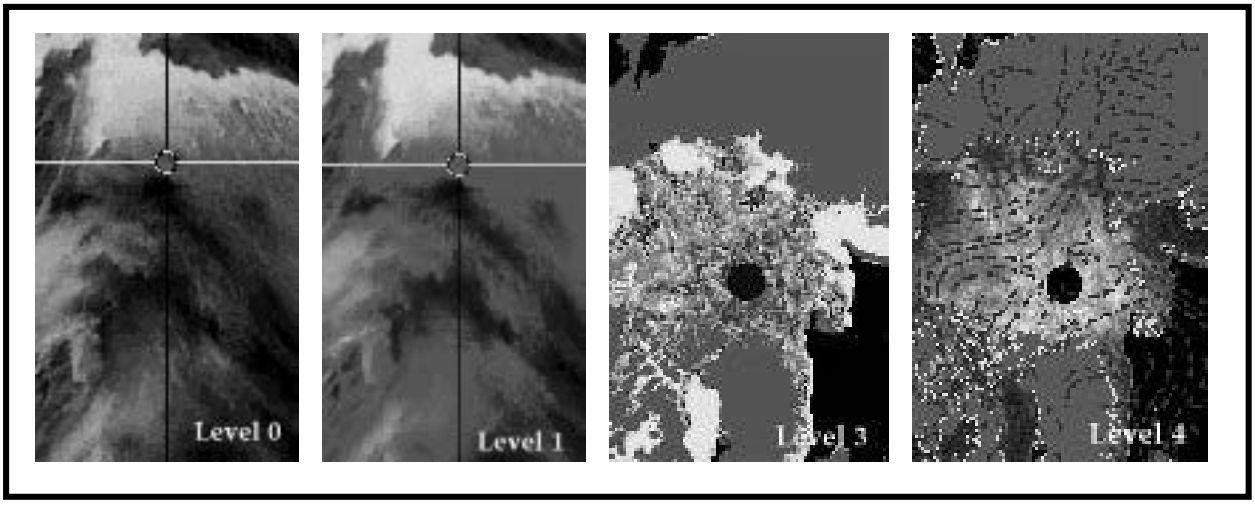

Figure 2. An Example of ECS Data Product Levels

Table 1. ECS Data Product Level Definitions

Level	Definition
0	Reconstructed, unprocessed instrument/payload data at full resolution; any and all communications artifacts (e.g., synchronization frames, communications headers, duplicate data removed)
1A	Reconstructed unprocessed instrument data at full resolution, time-referenced, and annotated with ancillary information, including radiometric and geometric calibration coefficients and geo-referencing parameters (e.g., platform ephemeris computed and appended but no applied to the Level 0 data)
1B	Level 1A data that have been processed to sensor units (not all instruments will have a Level 1B equivalent)
2	Derived geophysical variables at the same resolution and location as the Level 1 source data
3	Derived geophysical variables mapped on uniform space-time grid scales, usually with some completeness and consistency
4	Model output or results from analyses of lower-level data (e.g., variables derived from multiple measurements)

Subsystems

ECS is comprised of 13 subsystems, counting the Product Distribution System (PDS), as shown in Figure 3 and described in Table 2. More detailed information can be found in the *Release 7 Segment/Design Specification for the EMD Project* (305-EMD-001). Excluding the

Internetworking Subsystem (ISS), which ensures that the subsystems function as a whole, primary functions of the remaining subsystems can be grouped into the following four categories:

- **Data Ingest**. Ingest is accomplished by means of the Ingest Subsystem (INS), which interfaces with external applications and provides data staging capabilities and storage for an approximately 1-year buffer of Level 0 data so that reprocessing can be serviced from local storage. The number of external interfaces which ECS will have is potentially very large, and the interfaces can serve very diverse functions, such as high-volume ingest of level 0 data and low-volume ingest of data from field campaigns.
- **Data Processing**. Data processing is accomplished by means of the Data Processing Subsystem (DPS) for the science software and by capabilities for long and short term planning of science data processing, as well as by management of the production environment provided by the Planning Subsystem (PLS). Routine data processing and re-processing occur in accordance with the established production plans. In addition ECS provides on-demand processing, where higher-level products are produced only when there is explicit request.
- **Data Storage and Management**. Data storage and management is provided by the Data Server Subsystem (DSS), which can archive science data, search for and retrieve archived data, manage the archives, and stage data resources needed as input to science software or resulting as output from their execution. The Data Server Subsystem provides access to earth science data in an integrated fashion through an application programming interface (API) that is common to all layers.
- **Information Search and Data Retrieval**. Search and retrieval is accomplished by the science user interface functions in the Client Subsystem (CLS), and by information search support functions in the Data Management Subsystem (DMS).

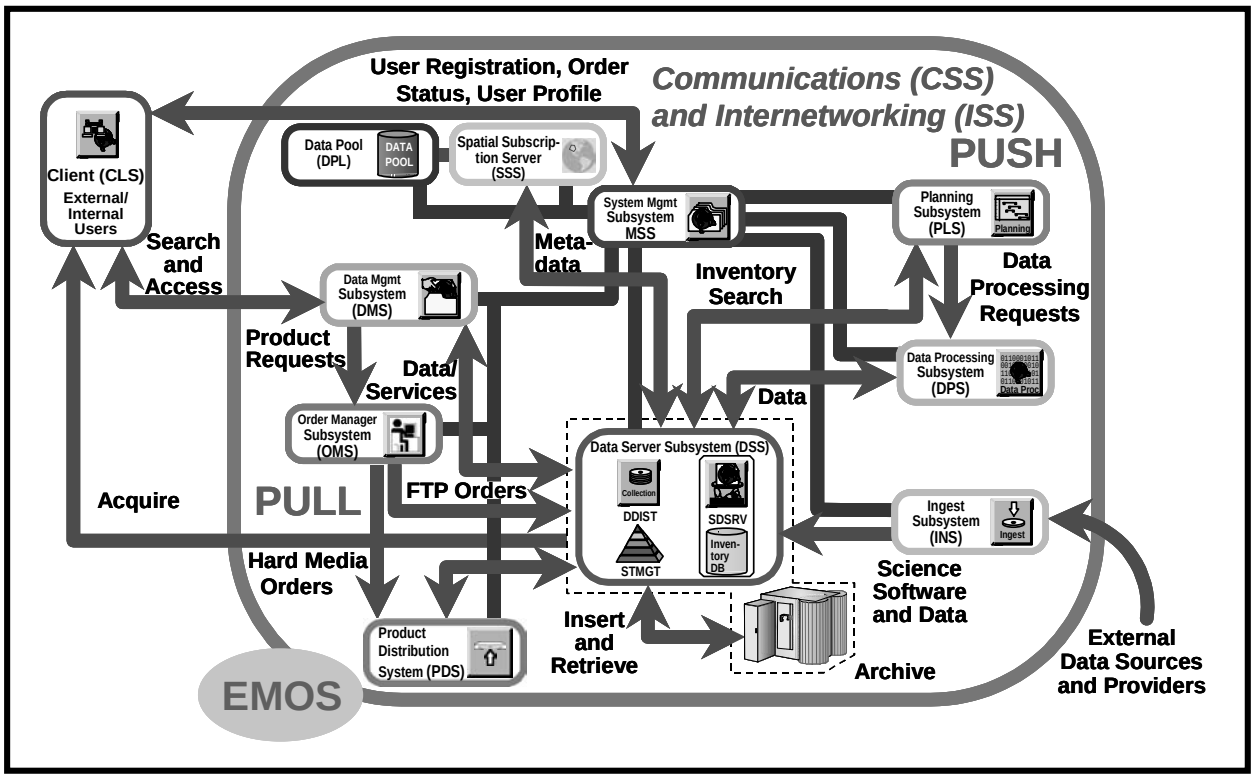

Figure 3. ECS Subsystem Interactions

Table 2. ECS Subsystem Functions

Subsystem		Functions
Client	CLS	The client part of the client-server access paradigm through graphical user interface and data/service access tools, as well as application program interface (API) libraries to ECS services
Internetworking	ISS	Provides networking services based on protocols and standards corresponding to the lower four layers of the Open Systems Interconnection (OSI) reference model: the transport layer's TCP and UDP protocol, the network layer's IP protocol, and the physical/data link layer's Ethernet, FDDI, and Gigabit Ethernet protocols.
Data Management	DMS	System wide distributed search and access services with multiple science discipline views of data collections and "one-stop shopping" with location transparent access to those services and data.
Data Server	DSS	Locally optimized search, access, archive and distribution services with a science discipline view of data collections and an extensible Earth Science Data Type and Computer Science Data Type view of the archive holdings
Ingest	INS	Clients for importing data (science products, ancillary, correlative, documents, etc.) into ECS data repositories (data servers) on an *ad hoc* or scheduled basis and deals with external system interfaces

Table 2. ECS Subsystem Functions

Planning	PLS	Pre-planning of routine/ad hoc/on-demand science data processing as well as management functions for handling deviations from the operations plans for individual DAAC sites
Data Processing	DPS	Hosts science algorithm software, performs science software integration and test, data processing, resource management and includes facilities and toolkits that offer true software portability across computing platforms
Management	MSS	Functions for system startup/shutdown, resource management, performance monitoring, error logging, system and science software configuration management, and resource accounting
Communications	CSS	The distributed computing infrastructure that enables intra- and inter-site communications between the subsystems. From a scientist user's perspective, the ECS infrastructure appears as services, and interfaces to services, which are displayed on the user's workstation desktop. This perspective is similar to the view of the World Wide Web (WWW) servers as seen through a local client such as Netscape.
Data Pool	DPL	Provides on-line access for browsing and FTP download of selected granules, metadata, and browse data.
Spatial Subscrip-tion Server	SSS	Permits creation and management of subscriptions for data distribution/ notification and Data Pool insert
Order Manager	OMS	Manages orders from EDG and other sources, distributing them to appropriate ECS services (e.g., SDSRV, PDS).
Product Distribution System	PDS	An ancillary system integrated with ECS to provide distribution of ECS products on hard media (8mm tape, CD ROM, DVD, DLT).

Databases

Custom Databases. The custom databases described in Table 3 encompass the majority of the ECS subsystem persistent data requirements. Other data requirements are met through the use of flat files, which are described below. With the exception of the PDS databases, which are implemented using Oracle, all custom databases are implemented using Sybase.

Table 3. ECS Custom Databases

Database Name	Document Number	DB Software	No. of Tables	Logical Categories
Science Data Server Subsystem (SDSRV)	311-EMD-004	Sybase	154	Database Version Information
				System Management Data
				Collection, Granule Metadata
				DAP Metadata
				Spatial Metadata
				Data Originator Metadata
				Granule Metadata
				Contact Metadata
				Collection Metadata
				Temporal Metadata
Planning and Data Processing Subsystems (PDPS)	311-EMD-003	Sybase	80	Database Version Information
				Planning Data
				Data Processing Data
Data Management Subsystem (DMS)	311-EMD-001	Sybase	63	Database Versioning
				Attribute/Term Definitions
				Collection Metadata
				Information Management
Storage Management and Database Distribution Subsystems (STMGT & DDIST)	311-EMD-005	Sybase	67	Database Version Information
				Data Distribution
				Archive Services
				Request Handling
				Server Configuration
				Cache Management
				Media Operations
				FTP Services
				Staging Disk Operations
				GR Cleanup
Ingest Subsystem (INS)	311-EMD-002	Sybase	25	Database Version Information
				Datatype Information
				Configuration Data
				Active Requests
				Validation Data
				Table Locking Information

Table 3. ECS Custom Databases

Database Name	Document Number	DB Software	No. of Tables	Logical Categories
Registry (REGIST)	311-EMD-008	Sybase	12	Database Version Information
				Security Information
				Registered Parameter Info
Systems Management Subsystem (MSS)	311-EMD-007	Sybase	19	Database Version Information
				Order Information
				Site Information
				Validation Data
				User Data
Subscription Server (SUBSRV)	311-EMD-006	Sybase	8	Database Version Information
				Subscription Information
				Event Information
NameServer (NM)	311-EMD-010	Sybase	2	Database Versioning
				NameServer
Product Distribution System (PDS)	311-EMD-009	Oracle	28	PDS Interface Server Order Data
				PDS Job Data
Data Pool (DPL)	311-EMD-013	Sybase	67	Collection Metadata
				Granule Metadata
				Insert Action Data
Order Manager Server (OMS)	311-EMD-011	Sybase	31	Queue/Status Information
				Request Information
				Intervention Information
Spatial Subscription Server (SSS)	311-EMD-012	Sybase	39	Database Version Information
				Subscription Information
				Event Information
				Action Information

Commercial Off-the-Shelf (COTS) Databases

Table 4 shows the most important databases resident in ECS COTS products. AutoSys is an application that supports data processing job scheduling and management. Tivoli is an integrated desktop that supports system administration, system monitoring, and performance and fault monitoring. Remedy provides a means of reporting, classifying, and tracking problems. XRP-II is a baseline manager used to maintain records of baselined operational system configurations. Database descriptions can be obtained by consulting the relevant vendor documentation.

Table 4. Primary ECS COTS Databases

Subsystem	COTS Product/Database Name	DB Software
PDPS	AutoSys	Sybase
MSS	Remedy	Sybase

Flat Files

A flat file is an operating system file that is read and written serially. Flat file data are fairly static and have no explicit relationship to other data in the enterprise. There are cases when the implementation of certain persistent data is better suited to a flat file than to a database, e.g., system configuration data, external interface data, log files. Flat files used in ECS are described in Table 5.

Table 5. ECS Flat Files

Database	Flat File Attributes			
	Usage	**Types**	**Formats**	**Descriptions**
SDSRV	Yes	UNIX flat file; ELF 32-bit MSB dynamic lib SPARC Version 1, dynamically linked	Variable length, Dynamic Link Library (DLL)	Log files, configuration files, template used to validate ESDTs on installation, uniquely named ESDT file descriptors, generic to ESDT-specific processing capabilities
PDPS	Yes	Text	ODL	Science metadata ODL file template
DMS	Yes	UNIX flat file	Variable length	Log files, configuration files
STMGT & DDIST	Yes	UNIX flat file	Variable length	Disk index files, staging data information, resource lists
INS	Yes	UNIX flat file	Variable length	Log files, configuration files, data delivery records
REGIST	No			
MSS	Yes	ASCII, binary	Single line records, one/two fields; EcAgEvent objects; MsAgMgmtHandle object; integers; string lists	Accountability component files, subagent component files
SUBSERV	No			
NM	No			
PDS	Yes	Text	ODL, Variable length	Production parameter files, status files, order data

Table 5. ECS Flat Files

Database	Flat File Attributes			
	Usage	**Types**	**Formats**	**Descriptions**
DPL	Yes	ASCII	Variable length	For Data Pool Access Statistics Utility, temporary storage of data to be exported to database
OMS	No			
SSS	No			

Resident Databases

Table 6 shows the custom and COTS databases that are resident at the SMC and the DAACs. Each resident database is individually installed and maintained.

Table 6. Resident ECS Databases

Databases	SMC	DAAC			
		GSFC	**EDC**	**LaRC**	**NSIDC**
Custom					
Science Data Server Subsystem (SDSRV)		✔	✔	✔	✔
Planning and Data Processing Subsystem (PDPS)		✔	✔	✔	
Data Management Subsystem (DMS)	✔	✔	✔	✔	✔
Storage Management and Database Distribution Subsystems (STMGT & DDIST)		✔	✔	✔	✔
Ingest Subsystem (INS)		✔	✔	✔	✔
Registry (REGIST [MSS])	✔	✔	✔	✔	✔
Systems Management Subsystem (MSS)	✔	✔	✔	✔	✔
Subscription Server (SUBSRV)		✔	✔	✔	✔
NameServer (NM)	✔	✔	✔	✔	✔
Product Distribution System (PDS)		✔	✔	✔	✔
Data Pool (DPL)		✔	✔	✔	✔
Order Manager Subsystem (OMS)		✔	✔	✔	✔
Spatial Subscription Server (SSS)		✔	✔	✔	✔
Replication Server System Database (RSSD)		✔	✔	✔	✔
COTS					
AutoSys (PDPS)	✔	✔	✔	✔	
Remedy (MSS)	✔	✔	✔	✔	✔

Replicated Databases. In the ECS design, the System Monitoring Center SMC database is the primary database for all records. This means that all user account requests, profile creation, and profile modification activities must take place at the SMC. Sybase replication capabilities automatically send committed database changes from the SMC to the other sites. This significantly decreases the elapsed time from when a change is made to when that change is propagated to other sites. Figure 4 illustrates replication for the User Profile database.

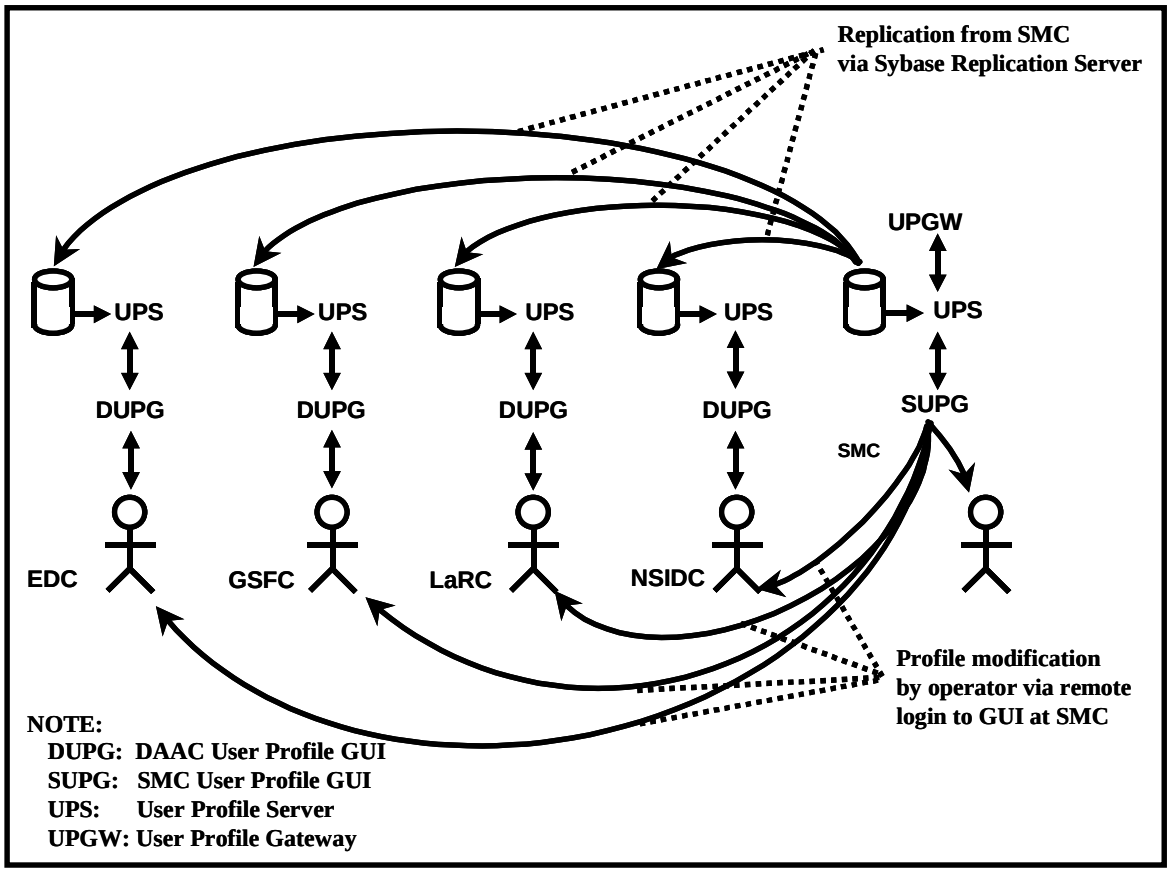

Figure 4. ECS Database Replication

Database Directory Locations

Locations of principal database components are shown in Table 7.

Table 7. Location of Principal Database Components

Name	Variant	Vendor	Principal Directory	Comments
Open Client	PC	Sybase	c:\windows\system	
Open Client/C	SGI	Sybase	/tools/sybOCv12.0.0	Just utilities, not libraries
Open Client/C	SUN	Sybase	/tools/sybOCv11.1.1	
Oracle Developer	SGI	Oracle	TBI	PDS only
Oracle Enterprise 8I	SGI	Oracle	TBI	PDS only/Oracle Forms 4.5 bundled
Replication Server	SUN	Sybase	/usr/ecs/OPS/COTS/sybase1151	
Replication Server Manager	SUN	Sybase	/usr/ecs/OPS/COTS/sybase1151	
SQL Server Monitor Client/Svr	SUN, SGI	Sybase	/usr/ecs/<mode>/COTS/sybase	At DAAC discretion/ required for launch
Spatial Query Server (SQS)	SGI	Autometrics	/usr/ecs/OPS/COTS/sqs_322/bin	
Sybase Adaptive Server	SUN, SGI		/usr/ecs/OPS/COTS/sybase1151	/usr/ecs/OPS/COTS/sybase_1151 is an acceptable install dir.
Sybase Adaptive Svr Enterprise	SUN, SGI	Sybase	/usr/ecs/OPS/COTS/sybase1151	/usr/ecs/OPS/COTS/sybase_1151 is an acceptable install dir.
Sybase Adaptive Svr Enterprise	SGI	Sybase	/usr/ecs/OPS/COTS/sybase_1193	
Sybase Adaptive Svr Enterprise	Sun	Sybase	/usr/ecs/OPS/COTS/sybase1151	/usr/ecs/OPS/COTS/sybase_1151 is an acceptable install dir.
Sybase Central	PC	Sybase	TBI	

ECS Database Management

ECS Database Management Model

The concept that forms the basis of ECS database management is described in *EOSDIS Core System Science Information Architecture* (FB9401V2). The ECS database management model is a variant of the ISO Data Management Reference Model (ISO 10032:1994). The variation is in the local (site) data management. The ISO reference model does not provide a structure for local data management. ECS defines local data management as being provided by a local information manager and a collection of data servers.

The model permits the following services and requests:

- *Client is a program requesting data management services.* A client can be an application program, such as a science algorithm, or a user interface, for example, an

interface for formulating database queries and displaying query results. The client may use a data dictionary or vocabulary to assist in the formulation of database requests.

- *Data Server is an instance of a service that is capable of executing data access, query, and manipulation requests against a collection of data*. Data server actually refers to a service provider at a logical level. The data server may actually be constructed from multiple physical servers implementing various aspects of the data server's functionality. Data servers use lower layers of data management services, which are described in the data server architecture, section 6.4. Note that a site may choose to provide access to the same data via several different data servers, perhaps supporting different data access and query languages.

- *Data Dictionary Service is* a service that manages and provides access to databases containing information about data. Each data object, data element, data relationship, and access operation available via data servers is defined and described in the dictionary databases. Data dictionaries are intended for access by users (e.g., to obtain a definition of a data item) and programs (e.g., to format a screen).

- *Vocabulary Service is* a service that manages and provides access to databases containing the definition of terminology, e.g., of words and phrases. Vocabularies are intended for access by users (e.g., to obtain the appropriate term given a meaning) and programs (e.g., to let a user identify the intended meaning of a term which has several alternative definitions).

The model identifies the following key data objects supporting these services:

- *Schema is* a formal description of the content, structure, constraints, and access operations available for or relevant to a database or a collection of databases. The reference model contains Data Server and Data Dictionary/Vocabulary schema.

The model provides for the following kinds of requests:

- *Query is a request formulated in a language offered (i.e., supported) by a data server.* It contains data search and access specifications expressed in terms defined either by the language itself, or in a schema offered by the corresponding server. In general, a query language is paired with a particular type of schema. For example, relational queries reference objects defined in a relational schema.

- *Data Access Request is a service request that invokes an operation offered by a data server on a data object or a set of data*. The object types and the operations that a given service offers for them are described in the schema used by the service.

In addition, Data Servers conform to the general interface requirements as prescribed by the Interoperability Architecture. For example, this means that Data Servers accept requests regarding the status and estimated cost of a query or data access request. The servers also use the Interoperability Services in the course of their interactions. For example, clients use request brokers in order to locate a data dictionary service. This use of interoperability services follows the general pattern defined in section 6.1, and is usually not explicitly mentioned in this section.

Database Management Implementation

Software

As previously described, ECS databases are primarily based on Sybase software. Only PDS uses Oracle software. Primary components include:

- **Sybase Adaptive Server Enterprise (ASE)**. ASE is an integrated set of software products for designing, developing and deploying relational database applications. It consists of a high-performance relational database management system (RDBMS), which runs database servers, and a collection of applications and libraries, which run on database clients. This arrangement, consisting of servers that are accessed by multiple clients over a network, forms the basis for Sybase's client/server architecture. ASE components are described in Table 8. An illustration of the main screen of Sybase Central, which can be used to perform many of the DBA procedures described in subsequent sections of this document, is shown in Figure 5.
- **Other Sybase Components:**
  - **Spatial Query Server (SQS)**. The Spatial Query Server (SQS) is a multi-threaded Sybase Open Query database engine that is used by the Science Data Subsystem (SDSRV). This product allows definition of spatial data types, spatial operators, and spatial indexing. SQS communicates with the Sybase SQL Server to process SDSRV requests to push and pull metadata. Both the SDSRV database server and the SQS server resides on the same SGI machine.
  - **Replication Server (RS)**. The Sybase Replication Server is used by the Management Subsystem (MSS) to allow all science users to register only once at their specified home DAAC and be able to request data from all DAAC sites. The RS uses replication definitions for tables at the primary database and subscriptions for tables at the replication sites. The replication definitions specify the location of the primary data while the subscription specifies the location of the replicate data. The replication definition and subscriptions are specified at the table or stored procedure level. The model attempts to prevent data inconsistencies that would be created by updating and replicating copies of the same data in two databases simultaneously. It stipulates that inserts, updates, and deletes to the data can only occur at the primary database, while select statements may occur at either the primary or the replicate database. The model assumes the enforcement of data ownership using custom developed database triggers, client code, or operational procedures.
- **Oracle Enterprise**. PDS uses the Oracle relational database management system software application to manage PDS order and job data. It also uses Oracle Forms Graphical User Interfaces (GUIs) permitting technicians to respond to problems with order processing, perform database maintenance, and monitor and control job processing. An Oracle Report Generator is used to create reports. The Oracle report modules reside on the PDS machine and use the SQL*Net connection to get any data

needed for the reports. The PDS operator interface module generates ASCII parameter files using Oracle Form's TEXT_IO package, which allows both reading and writing of operating system files.

Table 8. Sybase Adaptive Server Enterprise (ASE) Components

Type	Component	Description	Sub-Components and Features
Client	Sybase Central	A Windows application for managing Sybase databases. Helps manage database objects and perform common administrative tasks.	Connecting to, disconnecting from, and stopping servers
			Troubleshooting Adaptive Server problems
			Managing data caches
			Managing Adaptive Server physical resources
			Creating, deleting, backing up, and restoring databases
			Creating and deleting Adaptive Server logins, creating and deleting database users and user groups, administering Sybase roles, and managing object and command permissions
			Monitoring Adaptive Server performance data and tuning performance parameters
	Sybase Central Plug-Ins	Each server product is managed by a service provider plug-ins that coexists with other service providers in the Sybase Central framework.	ASE Plug-In
			SQS Plug-In
			Replication Server Manager (RSM). Provides the ability to manage, monitor, and troubleshoot most replication system components (primary and replicate database servers, Replication Servers, Replication Agents, and database gateways).
	Open Client		CS-Library, which contains a collection of utility routines used by all client applications.
			Client-Library and DB-Library, which contain a collection of routines that are specific to the programming language being used in an application
			Net-Library, which contains network protocol services that support connections between client applications and Adaptive Server.

Table 8. Sybase Adaptive Server Enterprise (ASE) Components

Type	Component	Description	Sub-Components and Features
			Utilities: isql – an interactive query processor that sends commands to the RDBMS from the command line. bcp – a program that copies data from a database to an operating system file, and vice versa. defncopy - a program that copies definitions of database objects that from a database to an operating system file and vice-versa.
Server	Adaptive Server (ASE)	Sybase's high-performance RDBMS	
	Backup Server(TM)	A server application that runs concurrently with Adaptive Server to perform high-speed on-line database dumps and loads.	
	Adaptive Server Monitor	Monitor Server	Allows capture, display, and evaluation of Adaptive Server performance data and tune Adaptive Server performance
		Historical Server	Writes the data to files for offline analysis

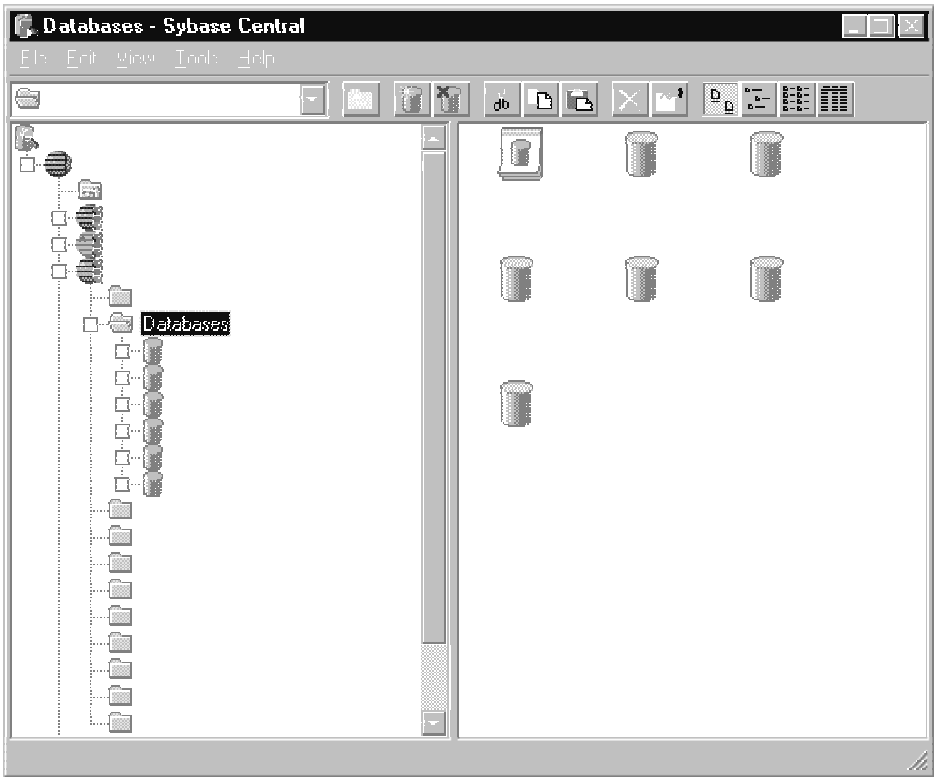

Figure 5. Sybase Central

Hardware, Software, and Database Mapping

Currently, ECS databases and database management systems reside on PCs (Sybase Central only) and SUN and SGI machines. Mapping documents for the DAACs are available on the web at http://cmdm.east.hitc.com/baseline, through the **Technical Documents** link. The hardware layout diagrams are available in documents 920-TD*x*-001, *Hardware-Design Diagram*. Hardware to software mappings are available in documents 920-TD*x*-002 *Hardware-Software Map*. The hardware database mappings are available in document 920-TD*x*-009 *DAAC HW Database Mapping*.

This page intentionally left blank.

ECS Database Administrator (DBA) Responsibilities

The ECS Database Administrator or DBA is the individual responsible for the installation, configuration, update, maintenance, and overall integrity, performance and reliability of ECS databases. In general, the DBA is concerned with the availability of the server, the definition and management of resources allocated to the server, the definition and management of databases and objects resident on the server, and the relationship between the server and the operating system. Basic DBA responsibilities include:

- Performing the database administration utilities such as database backup, maintenance of database transaction logs, and database recovery.
- Monitoring and tuning the database system (e.g., the physical allocation of database resources).
- Maintaining user accounts for the users from the external system. The DAAC database administrator creates user registration and account access control permissions in the security databases.
- Creating standard and *ad hoc* security management reports of stored security management data using the Sybase report generator.
- Working with EMD sustaining engineering and DAAC system test engineers to set up a test environment as needed.
- Working with the data specialist on information management tasks involving databases, data sets, and metadata management.
- Consolidating event reports into a site event history database for reporting activities to the SMC on a regular basis.
- Performing daily database synchronization.
- Administering the Replication Server System Database (RSSD).

DBA Tasks and Procedures

Basic DBA tasks and procedures are described in the following sections. Table 9 shows DBA tasks that have to be done on a regular basis and the section where they are addressed in this document.

Table 9. DBA Tasks Performed on a Regular Basis

Time Period	Task	Importance	Found In …
Daily	Capture database configurations	Absolutely necessary for database recovery if problems occur	Configuring Databases
Weekly	Monitor Sybase disk usage		Monitoring and Tuning Databases
	Clean up old files		
Monthly	Reboot		Starting and Stopping Servers
Before and After Installations	Run DbVerify scripts		Installing Databases and Patches

Starting and Stopping Servers

Starting Servers

A server process is started manually when a new server is installed or after a system outage. In order to perform the procedure, the DBA must know the database server name and be assigned a **sa_role** (see Establishing Database Security, below). The following procedures are applicable.

Starting SQL Servers

1 Log in to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press the **Return/Enter** key.

3 Log into the server on which the SQL Server Process is to be started by typing: **/tools/bin/ssh *ServerName***, then press the **Return/Enter** key.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Return/Enter** key.

- Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Return/Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press the **Return/Enter** key.

- A password prompt is displayed.

7 Enter ***sybase password***, then press the **Return/Enter** key.

- You are authenticated as yourself and returned to the UNIX prompt.

8 At the UNIX prompt, type **cd install** and then press the **Return/Enter** key.

9 At the UNIX prompt, type **./RUN_*servername* & *)***, then press the **Return/Enter** key.

10 At the UNIX prompt, type **showserver**, then press the **Return/Enter** key.

- This displays a listing of the SQL Server processes that are running.

Starting the SQL Backup Server

NOTE: This procedure should be done after SQL Server is up and running

1 Log in to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press the **Return/Enter** key.

3 Log into the server on which the SQL Server Process is to be started by typing: **/tools/bin/ssh *ServerName***, then press the **Return/Enter** key.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Return/Enter** key.

- Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Return/Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press the **Return/Enter** key.

- A password prompt is displayed.

7 Enter ***sybase password***, then press the **Return/Enter** key.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type **cd install** and press the **Return/Enter** key.

9 Type, **./RUN_*backupservername* &** and press the **Return/Enter** key.

NOTE: You may have to hit the **Return/Enter** key one more time

Starting the SQL Monitor Server

NOTE: This procedure should be done after the SQL Server is up and running

1 Log in to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press the **Return/Enter** key.

3 Log into the server on which the SQL Server Process is to be started by typing: **/tools/bin/ssh** ***ServerName***, then press the **Return/Enter** key.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Return/Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Return/Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press the **Return/Enter** key.

- A password prompt is displayed.

7 Enter ***sybase password***, then press the **Return/Enter** key.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type **cd SMS11.x/bin** and press the **Return/Enter** key.

9 Type **./monserver –S*server-name* –M*monitor-server-name* –Usa &** and press the **Return/Enter** key.

Stopping Servers

Servers are stopped by the DBA when the system needs to be brought down for maintenance. Servers are brought down in an order that is the reverse of the start-up order. The **shutdown** command issued from within isql shuts down the server where you are logged in. It allows for the completion of any current processes and blocks the start of any new processes before the server is shutdown.

When you issue a **shutdown** command, the server:

- Disables all logins except the sa's.
- Performs a checkpoint in each database, moving changed pages from memory to disk.
- Waits for currently executing SQL statements or procedures to finish.

Adding a **nowait** option to the command immediately terminates all processes and shuts down the server. The **nowait** option should be used sparingly because it may cause data loss and complicate recovery.

Stopping the SQL Monitor Server

NOTE: This procedure should be done before the SQL Server is shutdown

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press the **Return/Enter** key.

3 Log into the server on which the SQL Server Process is to be stopped by typing: **/tools/bin/ssh *ServerName***, then press the **Return/Enter** key.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Return/Enter** key.

- Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Return/Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press the **Return/Enter** key.

- A password prompt is displayed.

7 Enter ***sybase password***, then press the **Return/Enter** key.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type **setenv SYBASE *sybase-directory-path*** then press the **Return/Enter** key.

9 Type **setenv DSQUERY *sql-servername*** then press the **Return/Enter** key.

10 Type **$SYBASE/bin/isql –Usa** and press the **Return/Enter** key.

11 Type the ***sa-password*** when prompted for it and press the **Return/Enter** key.

12 Type **sms_shutdown** at "**1>**" prompt and press the **Return/Enter** key.

13 Type **go** at "**2>**" prompt and press the **Return/Enter** key.

Stopping the SQL Backup Server

NOTE: This procedure should be done before shutting down the SQL Server

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press the **Return/Enter** key.

3 Log into the server on which the SQL Server Process is to be stopped by typing: **/tools/bin/ssh *ServerName***, then press the **Return/Enter** key.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Return/Enter** key.

- Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Return/Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press the **Return/Enter** key.

- A password prompt is displayed.

7 Enter ***sybase password***, then press the **Return/Enter** key.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type **setenv SYBASE *sybase-directory-path*** then press the **Return/Enter** key.

9 Type **setenv DSQUERY *sql-servername*** then press the **Return/Enter** key.

10 Type **set path = ($path $SYBASE/bin $SYBASE/install $SYBASE)** then press the **Return/Enter** key.

NOTE: GDAAC implementation has a sybsetup.csh file (type source sybsetup.csh).

11 Type, **isql –U<username>** and press the **Return/Enter** key.

12 Type the ***password*** when prompted for it and press the **Return/Enter** key.

13 Type **shutdown SYB_BACKUP** at the "**1**>" prompt and press the **Return/Enter** key.

14 Type **go** at the "**2**>" prompt and press the **Return/Enter** key.

Stopping the SQL Server

1 Use **shutdown** to bring the server to a halt.

- This command can only be issued by the Sybase System Administrator (sa).
- If you do not give a server name, shutdown shuts down the SQL Server you are using.
- Syntax:

  1> shutdown [backup_server_name]] [with] [wait] [with nowait]

  2> go

2 To see the names of the backup servers that are accessible from your SQL Server, execute **sp_helpserver**.

- Use the value in the name column in the shutdown command.
- You can only shut down a Backup Server that is:
  - Listed in **sysservers** on your SQL Server
  - Listed in your local interfaces file

Creating Database Devices and Logical Volumes

Database Devices

Database devices store the objects that make up databases. The term "device" can refer to either a physical or a logical device. Logical devices include any portion of a disk, such as a *partition* or a *file* in the file system used to store databases and their objects. A database device can be created when the System Administrator determines that new disk space is available for database use or as part of the recovery process. The System Administrator makes a request to the Data Base Administrator (DBA), who creates the new database device and notifies the System Administrator when the device has been created.

In order to create a new device, the DBA must have the following:

- The name of database device to be created
- A physical device on which to place database device
- The device size in megabytes
- The name of the mirror device, if one is in effect

*The creation of a database device is the mapping of physical space to a logical name and virtual device number (**vdevno**) contained in the master database*. The **disk init** command is used to initialize this space. **Disk init** syntax is:

name = "device_name"

physname = "physicalname"

vdevno = virtual_device_number

size = number_of_blocks

[vstart = virtual_address

cntrltype = controller_number]

Before you run **disk init**, see the server installation and configuration guide for your platform for information about choosing a database device and preparing it for use with the server. You may want to repartition the disks on your computer to provide maximum performance for your databases. **Disk init** divides the database devices into allocation units of 256 2K pages, a total of 1/2MB. In each 256-page allocation unit, the **disk init** command initializes the first page as the allocation page, which will contain information about the database (if any) that resides on the allocation unit.

Once initialization is complete, the space described by the physical address is available to the server for storage and a row is added to the **sysdevices** table in the master database. The initialized database can be:

- Allocated to the pool of space available to a user database
- Allocated to a user database and assigned to store a specific database object or objects
- Used to store a database's transaction logs

NOTE: Unless you are creating a small or non-critical database, always place the log on a separate database device.

After you run the **disk init** command, be sure to use **dump database** to dump the master database. This makes recovery easier and safer in case master is damaged. If you add a device and fail to back up master, you may be able to recover the changes with **disk reinit**.

Creating a Database Device

---

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press the **Return/Enter** key.

3 Log into the server on which the new database device is to be created by typing: **/tools/bin/ssh <*ServerName*>**, then press the **Return/Enter** key.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Return/Enter** key.

- Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Return/Enter** key.

NOTE: For each database device to be created, perform Steps 6 through 9.

6 At the UNIX prompt, type **cd server.devices** then press the **Return/Enter** key.

- This places you in the directory that contains all the scripts used to create database devices.
- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure.

7 At the UNIX prompt, type **cp template.sql *DeviceName*.sql** then press the **Return/Enter** key.

- This creates a new SQL script file into which you will type the appropriate commands to create the database device.

8 Using the text editor of your choice, edit ***DeviceName*.sql** and make changes to information enclosed in brackets (including the brackets) as appropriate:

- An example of an **add_devices.sql** file for creation of a database device is shown in Figure 6.

```
/*******************************************************/
/* name: [add_devices.sql]                             */
/* purpose:                                            */
/* written:                                            */
/* revised:                                            */
/* reason:                                             */
/*******************************************************/
disk init name = [device name],
physname = "/dev/[device name]",
vdevno = [#],
size = [size]
go
sp_helpdevice [device name]
go
```

Figure 6. Example of an add_devices.sql File for Creation of a Database Device

- In the area delimited by **/* */**, enter an appropriate description of the script including the file name, date written and person who wrote the script, its purpose, and any other information deemed appropriate. Be sure to enclose each line of the comment between **/* */**.
- The **disk init name** is the ***DeviceName*** is used in Step 8 above. You may not use spaces or punctuation except the underscore character (_) in ***DeviceName***. Remember that the name you assign is case-sensitive. Be sure there is a comma after ***DeviceName***
- The physname is the *FullPath_to_DeviceName*. Be sure to enclose *FullPath_to_DeviceName* in double quotes. Be sure to place a comma after ***FullPath_to_DeviceName*** but outside the double quotes.

- The **vdevno** is the ***VirtualDeviceNumber***. **vdevno** is a unique identifying number for the database device. It must be unique among all the devices used by SQL Server and is never reused. Device number 0 represents the device named **d_master** that stores the system catalogs. Legal numbers are between 1 and 255, but the highest number must be one less than the number of database devices for which your system is configured. For example, for a system with the default configuration of 10 devices, the legal device numbers are 1-9. The default of 10 devices can be changed using **sp_configure**. For the GDAAC implementation, the next vdevno is located in the /usr/dba0x/server.devices/next.vdevno file, it should be incremented when used. If this file does not exist, the next available number can be determined by looking at the output from the **sp_helpdevice** command and selecting the next number in sequence. If you use an existing number, Sybase will return the message, "device activation error."
- The **size** is the ***DeviceSize*** in blocks. To compute the number of blocks, multiply the device size in megabytes by 512; e.g., a 1,000 Mb device has 512,000 blocks

9 After the changes have been made, save the file according to the rules of your text editor.

10 At the UNIX prompt, type **isql –U*username* -S*ServerName* -i*DeviceName*.sql -o*Device Name.out*** then press the **Return/Enter** key.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***DeviceName.sql*** is the name of the script file you created in step 8.
- ***DeviceName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.)
- The system will prompt you for a password.

11 At the **Password:** prompt, type the **<*password*>** then press the **Return/Enter** key.

- When the UNIX prompt is again displayed the process is complete.

12 At the UNIX prompt, type **more *DeviceName.out***, then press the **Return/Enter** key.

- This allows you to view the ***DeviceName.out*** file to confirm that the device has been created or to check for device creation errors.
- A sample of a completed database device creation script is shown in Figure 7.

```
/*******************************************************/
/* name: test_dev.sql                                  */
/* purpose: allocate 3Mb device for testing            */
/* written: 12/18/97                                   */
/* revised:                                            */
/* reason:                                             */
/*******************************************************/
disk init name = test_dev,
physname                                                =
     "/usr/ecs/Rel_A/COTS/sybase/studentdevices/test_dev.dat",
vdevno = 15,
size = 1536
go
sp_helpdevice test_dev
go
```

Figure 7. Completed Database Device Creation Script

Logical Volumes

Logical volume disk driver (XLV) devices provide access to disk storage as logical volumes. *A logical volume is an object that behaves like a disk partition, but its storage can span several physical disk devices*. Using XLV, disks can be concatenated disks to create larger logical volumes, stripe data across disks to create logical volumes with greater throughput, and plex (or mirror) disks for reliability. In addition, XLV allows volume configurations changes to be made volumes are actively being used as a filesystem.

The geometry of logical volumes (e.g., the disks that belong to it, how they are put together, etc.) is stored in the disk labels of the disks that belong to the logical volumes. When the system starts up, the utility **xlv_assemble(1M)** scans all the disks on the system and automatically assembles them into logical volumes. **xlv_assemble(1M)** also creates any necessary device nodes.

Device numbers range from 0 to one less than the maximum number of logical volume devices configured in the system. This is 10 by default. This number can be changed by rebuilding a kernel with **lboot(1M).** There is a kernel driver, referred to as **xlv**, and some daemons for the logical volume devices. The driver is a pseudo device not directly associated with any physical hardware; its function is to map requests on logical volume devices into requests on the underlying disk devices. The daemons take care of error recovery and dynamic reconfiguration of volumes.

XLV allows the DBA to work with whole volumes and pieces of volumes. Pieces of volumes are useful for creating and reconfiguring volumes in units that are larger than individual disk partitions.

Each volume consists of up to three subvolumes. An **xfs(4)** filesystem usually has a large data subvolume in which all the user files and metadata such as inodes are stored and a small log subvolume in which the filesystem log is stored. For high-performance and real-time applications, a volume can also have a real-time subvolume that contains only user files aligned at configurable block boundaries. Guaranteed rate I/O can be done to real-time subvolumes.

Converting Raw Partitions into XLV

---

1 Back up all databases.

2 BCP the **syslogins** table from master.

3 bcp master..syslogins out file.out -Usa -P -c

4 bcp master..sysloginroles out file.out -Usa -P -c

5 Save all information regarding the **sysdevices** and database options by executing the following command: **sp_helpdb db_name -- each database sp_helpdevice sp_helpdb**

6 Shutdown the backup and sql servers.

7 After the partitions have been updated to XLV, **chown** the sybase disks to sybase:users. Also, change the **/etc/init.d/sybase** script.

8 Clean up some old Sybase files from **$SYBASE/devices** and **$SYBASE/install**.

9 Execute **sybinit** to initialize a NEW sql server/backup (same server name).

10 Initialize the **sybsystemprocs** database.

11 Change sa password. **(sp_password NULL,newpass**)

12 Increase the number of devices. (**sp_configure "number of devices",20**)

13 Alter the database for master and tempdb.

14 Change **sysservers** to name the server as well as the new backup server. **sp_addserver nodename_srvr,local,nodename_srvr** , **sp_addserver SYB_BACKUP,local,nodename_backup**

15 Execute scripts to initialize the devices and to create databases (for load).

16 Load database dumps.

17 Verify that database data and log are not on same device. If on same device, update **sysusages** by deleting the log that uses he data device and then update the size of the data device to the full size (plus the one used by the log).

18 Uncompress the load file manually.

19 Online the database.

20 BCP in the **syslogins** table.

21 BCP in the **sysloginroles** table.

22 Change the DB dbo.

23 Reset all DB options.

24 Dump all databases.

The backout procedure is executed only if it is necessary to restore the old SQL server because of a XLV conversion failure.

Backout Procedure

---

1 After restoring the sybase raw disk partitions from XLV partitions, change the ownership of the devices to **sybase:user**. This step has to be executed by someone with root access.

2 Login as sybase and cd to the **$SYBASE/install** directory.

3 Execute **sybinit** and configure a new SQL server.

- Use the same configurations that were used from original SQL server, i.e., port numbers.

4 Once you have configured a new server, get into ISQL and execute the following:

- Change sa password.
- Alter database master to same size as before.
- Modify the sysservers with the sql and backup names.
- Shutdown the SQL server and bring it up in single user mode (**-m** option).
- Uncompress the master database dump (do not load at this time).
- Change directory to **/usr/ecs/OPS/COTS/sybase/devices** and copy the old **sybsystemprocs.dat** to the new one.
- Load the dump of the master database.
  - After the load, it would automatically shut down the SQL server, so start it up as normal and not as single user.

5 If your databases do not recover at this point, mark them as suspect.

6 Check your devices and load each database from the dumps according to the following steps:

- Check all the devices and make sure that they exist.
- Drop the databases using **dbcc repair** (**dbname,dropdb**) option.

- Re-create the databases with a **for load** option.
- Load each of the databases and do an online command after
- Verify that your **db** options were set properly as well as all the users on each database

7 Once everything is back to its original state, do a complete backup of all your databases.

Installing Databases and Patches

Custom Databases

The ECS Assistant (see Figure 8) is a custom application that simplifies the process of installing, testing and managing ECS software. The **Subsystem Manager** screen, shown in Figure 8, is used in the operational environment. The **Database** option is used to install, drop, patch, and update subsystem-specific databases. The **Install** option is used to install ECS custom software in a particular mode. The **Configuration** option is used to create CFG, ACFG and PCFG files for selected components. The **Stage Area Installation** option is used to input the staging location where the delivered software is stored. The **View Task Output** option is used to view results as the specified task is executing. A detailed description of ECS Assistant use can be found in 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*.

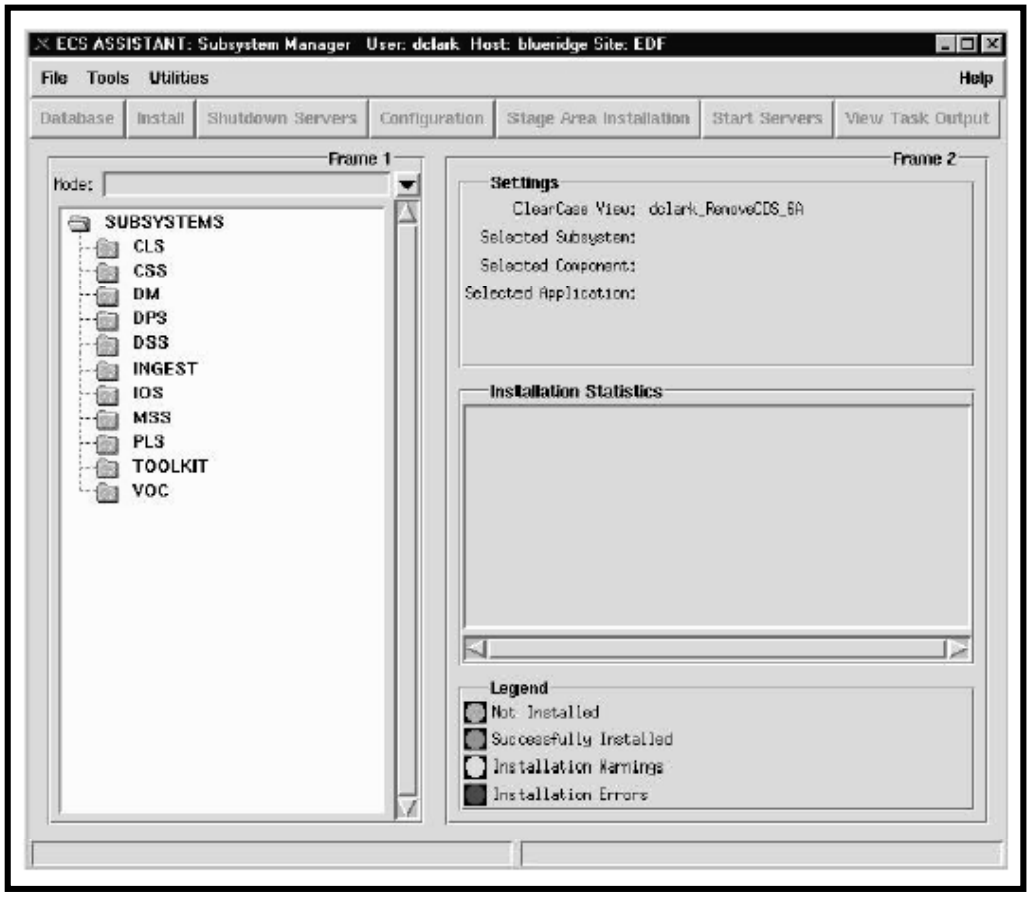

Figure 8. ECS Assistant

Example of a Database Build Procedure

To perform a database build follow the steps included in the documentation (e.g., release notes) that specifies performing the build. The following steps are an example of a database that was built and populated with data from the version of the database that the new database was replacing. Unless otherwise specified, the scripts that were run are located in the **/usr/ecs/<MODE>/CUSTOM/dbms/<SUBSYSTEM>** directory.

Perform a Database Build (Example)

1 Create new db logins for new servers.

- For example:

  # EcDsStDb6ALogins <MODE> <dbo_ user_ name> <dbo_ user_ password> <server_ name> <DBNAME>

2 Run the database verification script on the existing database.

- For example:

  # EcStDbVerify5B <MODE> <USERNAME> <SERVER> <DBNAME>

- The script is located in the **/usr/ecs/<MODE>/CUSTOM/dbms/COM/DBAdmin** directory. The output from this script is located in **/usr/ecs/<MODE>/CUSTOM/logs** and is named **EcStDbVerify5B.<DBSERVER>.<DBNAME>.<DATETIME>**.

3 Bulk copy data out of the existing database:

- For example:

  # EcDsStBulkCopyOut5B <MODE> <USERNAME> <PASSWORD> <SERVER> <DBNAME>

- The output from this script is saved as **/usr/ecs/<MODE>/CUSTOM/logs/EcDsStBulkCopy.log**.

4 Build the database using **ECS Assistant**.

- Using the **SubSystem Manager**, execute the **DbBuild** function to build the database.
- Enter the appropriate parameters in the **Database Script Parameters** dialogue box.

5 Execute the **EcDbDDMUpdateVersionTable.sql** script file that is under the directory **/usr/ecs/{MODE}/CUSTOM/dbms/COM/DBAdmin**.

- The parameters to be passed are **MODE USERNAME DSQUERY DBNAME**.
- You will be prompted for the password.

6 Review the file **EcMsDbBuild.log** in **/usr/ecs/<MODE>/CUSTOM/logs** for any error or warning messages.

7 Have the DBA verify the database changes through the **DbDesc** script for this subsystem.

8 Verify that the database version identifiers are correct.

- For example:

  # isql -S <server_ name> -U <db_ user_ name> -P <db_ user_ password>
  # use <database>_< MODE>
  # go
  # select * from EcDbDatabaseVersions where EcDbCurrentVersionFlag=" Y"
  # go

9 Run a database transition upgrade script.

- For example:

  # EcDsStDbSybaseUpgrade. ksh <MODE> <USERNAME> <PASSWORD> <SERVER> <DBNAME>

10 Run a script to create all of the new tables required for transition into the new database.

- For example:

  # EcDsStCreate5BTranTables <USERNAME> <PASSWORD> <SERVER> <DBNAME>

11 Bulk copy data in the old tables into the new database.

- For example:

  # EcDsStBulkCopyIn5B <MODE> <USERNAME> <PASSWORD> <SERVER> <DBNAME>

- The output from this script is saved as **/usr/ecs/<MODE>/CUSTOM/logs/EcDsStBulkCopy.log**.

12 Run the Transition Script.

- For example:

  # EcDsStTransition5Bto6A <MODE> <USERNAME> <PASSWORD> <SERVER> <DBNAME>

NOTE: You will need to have the secure shell passphrase and install login password ready to enter when prompted.

- The output from this script is saved as **/usr/ecs/<MODE>/CUSTOM/logs/EcDsStTransition.log**.

13 Run the database verification script (script is located in the **/usr/ecs/<MODE>/CUSTOM/dbms/COM/DBAdmin** directory).

- For example:

  # EcStDbVerify6A <MODE> <USERNAME> <SERVER> <DBNAME>

- The output from this script is saved as **/usr/ecs/<MODE>/CUSTOM/logs/EcStDbVerify6A.<DBSERVER>.<DBNAME>.<DATETIME>**.

14 Check the differences between the database verification script output logs for the old database and the new database:

- For example:

  # diff -w EcStDbVerify5B.<DBSERVER>.< DBNAME>.<DATETIME> EcStDbVerify6A.<DBSERVER>.<DBNAME>.<DATETIME>

15 Compare the output of this statement with the differences expected based on information in the reference document.

16 Run a script to remove all tables required for transition into the new database:

- For example:

  # EcDsStDrop5BTranTables <USERNAME> <PASSWORD> <SERVER> <DBNAME>

Example of a Database Patch Procedure

To install a database patch follow the steps included in the documentation (e.g., release notes) that specifies installing the patch. To install database patches, perform the following steps for all subsystems/components and then perform the appropriate subsystem/component-specific procedures.

Install a Database Patch (Example)

---

1 Verify the current version of the database being patched.

 # isql -S <server_ name> -U <db_ user_ name> -P <db_ user_ password>
 # use <db_ name>[_< MODE>]
 # go
 # select * from EcDbDatabaseVersions where EcDbCurrentVersionFlag=" Y"
 # go

2 Compare the current database version against the appropriate version listed in the table provided in the patch instructions.

- If the current version is greater than or equal to the appropriate version listed in the table, continue with the database patch.
- If the current version is less that the appropriate version listed in the table, stop and patch the deficient database.

3 From the **ECS Assist Subsystem Manager** select the appropriate mode, subsystem, and component from the main window.

4 From the **ECS Assist Subsystem Manager** select **DbPatch** from the **Database** menu.

- A **File Selection** window appears.

5 From the **ECS Assist Subsystem Manager** (in the **File Selection** window) select **.dbparms** and **OK**.

6 Follow subsystem-specific installation instructions included in the documentation (e.g., release notes) that specifies installing the patch to complete the database patch process.

- For example:

  1 Logon to the host where the subsystem database package is installed.

  2 Start ECS Assist's Subsystem Manager, select the appropriate mode, subsystem, and component.

  3 Select **DbPatch** from the **Database** menu. A **File Selection** window appears.

  4 In the "**File Selection**" window, select "**.dbparms**" and then "**Ok**". The "**Configurable Database Parameters**" dialog box appears.

  5 Verify the patch number (referring to the table in the patch instructions). If it is not correct, correct it, enter the required information and select **Ok**.

COTS Databases

AutoSys, Tivoli, and Remedy have Sybase databases. COTS installation is performed by the auto-install applications in the **/usr/ecs/OPS/COTS** directory. The results of the installation are stored in the appropriate subdirectories, e.g., an **autosys_install.scr** file located in the AutoSys home directory (**/use/ecs/OPS/COTS/autosys**).

This page intentionally left blank.

Configuring Databases

Configuration Parameters

Configuration parameters are user-defined settings that control various aspects of a database server's behavior. The server supplies default values for all configuration parameters. ECS requires the customization of some of the configuration parameters (see *SYBASE SQLServer 11.0.x ALL DAAC Database Configurations,* 910-TDA-021).

Configuration parameters are grouped according to the area of server behavior that they affect. This makes it easier to identify all parameters that may need to be tuned in order to improve a particular area of SQL Server performance. The groups are:

- Backup/Recovery.
- Cache Manager.
- Disk I/O.
- General Information.
- Languages.
- Lock Manager.
- Memory Use.
- Network Communications.
- Operating System Resources.
- Physical Memory.
- Processors.
- SQL Server Administration.
- User Environment.

While each parameter has a primary group to which it belongs, many have secondary groups to which they also belong. For instance, the parameter number of remote connections belongs primarily in the Network Communications group, but also secondarily in the SQL Server Administration group and the Memory Use group. This reflects the fact that some parameters have implications for a number of areas of SQL Server behavior.

The configuration parameters are divided between two tables:

- Sybase Configuration Parameter Table.
- DAAC-Specific Configuration Parameter Table.

The Sybase Configuration Parameter Table groups the configuration parameters by functionality. It provides a list of configurable parameters under each functional group, and provides the ECS recommended values for these parameters:

The following type of information is captured in the Sybase Configuration Parameter Table:

- Group Name.
- Group Description.
- Parameter Name (for each group).
- Parameter Description.
- Command Line.
- Status (Static or Dynamic parameter).
- ECS Recommended Value.
- Range (Minimum and maximum parameter value).
- Display Level (Basic, Intermediate, Comprehensive).
- Impact (Brief description of type of impact when a parameter value is configured).
- Controlling Authority (Entity responsible for controlling changes to a configuration parameter).
- Comments (This will include suggestions and examples).

Configuration parameters can be set or changed in one of two ways:

- By executing the system procedure **sp_configure** with the appropriate parameters and values.
- By hand-editing your configuration file and then invoking **sp_configure** with the configuration file option.

Configuration parameters are either dynamic or static. *Dynamic parameters go into effect as soon as **sp_configure** is executed. Static parameters require the SQL server to reallocate memory and take effect only after SQL Server has been restarted.*

The roles required for using **sp_configure** are as follows:

- Any user can execute **sp_configure** to display information about parameters and their current values.
- Only System Administrators and System Security Officers can execute **sp_configure** to modify configuration parameters.
- Only System Security Officers can execute **sp_configure** to modify values for allow updates, audit queue size and remote access.

The system procedure **sp_configure** displays and resets configuration parameters. One can restrict the number of parameters **sp_configure** displays by using **sp_displaylevel** to set the display level to one of the three values: Basic, Intermediate, or Comprehensive.

Figure 9 shows a sample **sp-configure** output. **Name** column is the description of the variable. The **minimum** column is the minimum allowable configuration setting**. Maximum** is the theoretical maximum value to which the configuration option can be set. The actual maximum value is dependent on the specific platform and available resources to the SQL server. **Config_value** reflects the current system default values. **Run_value** reflects the values the system is currently using, which may be different from the default values.

name	minimum	maximum	config value	run value
recovery interval	1	32767	0	5
allow updates	0	1	0	0
user connections	5	2147483647	0	25
memory	3850	2147483647	0	5120
open databases	5	2147483647	0	12
locks	5000	2147483647	0	5000
open objects	100	2147483647	0	500
procedure cache	1	99	0	20
fill factor	0	100	0	0
time slice	50	1000	0	100
database size	2	10000	0	2
tape retention	0	365	0	0
recovery flags	0	1	0	0
nested triggers	0	1	1	1
devices	4	256	0	10
remote access	0	1	1	1
remote logins	0	2147483647	0	20
remote sites	0	2147483647	0	10
remote connections	0	2147483647	0	20
pre-read packets	0	2147483647	0	3
upgrade version	0	2147483647	1002	1002
default sortorder id	0	255	50	50
default language	0	2147483647	0	0
language in cache	3	100	3	3
max online engines	1	32	1	1
min online engines	1	32	1	1
engine adjust interval	1	32	0	0
cpu flush	1	2147483647	200	200
i/o flush	1	2147483647	1000	1000
default character set id	0	255	1	1
stack size	20480	2147483647	0	28672
password expiration interval	0	32767	0	0
audit queue size	1	65535	100	100
additional netmem	0	2147483647	0	0
default network packet size	512	524288	0	512
maximum network packet size	512	524288	0	512
extent i/o buffers	0	2147483647	0	0
identity burning set factor	1	9999999	5000	5000
allow sendmsg	0	1	0	0
sendmsg starting port number	0	65535	0	0

Figure 9. Example of sp_configure Output

SQL Server configuration can be done either interactively by using **sp_configure** or non-interactively by invoking sp_configure with a configuration file. Configuration files can be used for several reasons:

- To replicate a specific configuration across multiple servers by using the same configuration file.
- To use a configuration file as a baseline for testing configuration values on your server.
- To use a configuration file to do validation checking on parameter values before actually setting the values.
- To create multiple configuration files and to switch between them as your resource needs change.

Previous releases of SQL Server required that reconfigure be executed after executing **sp_configure**. This is no longer required. The reconfigure command still exists, but it no longer has any effect. Note: If you have scripts that include **reconfigure**, you should change them at your earliest convenience.

Configuration files are not automatically backed up when you back up your master database. As they are part of the operating system files, they should be backed up in the same way other operating system files are backed up.

In order to perform the following procedure, the DBA must have obtained the following information:

- Name of server to be configured
- New values for configuration variables.

To set SQL Server configuration variables, the DBA must have **sa_role** (SQL Server) permissions. To set SQL Server configuration variables **allow updates**, **audit queue size**, **password expiration interval**, or **remote access**, **sso_role** (SQL Server) is also required.

Some parameter values take effect as soon as you reset the value. Others, which involve memory reallocation, do not change until you reset the value and then reboot SQL Server.

Configuring the SQL Server

---

1 Log into the server that will be reconfigured by typing: **telnet *ServerName*** or **rlogin *ServerName***, then press **Return**.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword*** then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

5 Begin the SQL session by typing at the UNIX prompt **isql -S*ServerName*** then press **Return**.

6 Type **sp_configure** then press **Return**.

7 Type **go**, then press **Return.**

- A list of the configurable parameters, their maximum and minimum values, the current setting and the default values is displayed.

8 After determining which parameter(s) to reset, type: **sp_configure "*ParameterName*"**, ***NewValue***, then press **Return**.

- ***ParameterName*** is the name from the list displayed in Step 7 above.
  - Be sure to enclose the name in double quotes.
- ***NewValue*** is the numeric value that you want to assign to ***ParameterName***.

9 Type **go**, then press **Return.**

10 Repeat Steps 7 through 9 for each parameter you want to reconfigure.

11 When complete, type **quit** at the ISQL prompt, then press **Return**.

- You are returned to the UNIX prompt.

Configuration Parameters and the Configuration Registry

There are many configurable parameters associated with ECS software. Some of them are set by default to values that may be appropriate for most operating conditions. Others may be set to values that may or may not be appropriate for the requirements of operations at a particular DAAC. Some parameters may be changed using ECS Graphical User Interfaces (GUIs) specifically designed to monitor and control functions related to particular subsystems. Others may require changes to a configuration file or database.

NOTE: Before changing any configuration parameter, make certain either that it is not under configuration control or that you have obtained any necessary approval specified in local Configuration Management policies and procedures.

Configuration Registry

ECS configuration parameters are managed by the Configuration Registry. The Configuration Registry Server provides a single interface to retrieve configuration attribute-value pairs for ECS

Servers from the Configuration Registry Database, via a Sybase Server. The Configuration Registry Server maintains an internal representation of the tree in which configuration attribute-value pairs are stored. General configuration parameters used by many servers are stored in higher nodes in the tree. Parameters specific to a single ECS Server are contained in the leaf nodes of the tree. ECS provides a script tool to load the Configuration Registry database from data in configuration files. This loading is a one-time event to populate the Registry database with the information contained in **.cfg** files. Once the Configuration Registry is loaded, if the configuration files are moved or otherwise made inaccessible to the software, the software goes to the Configuration Registry to obtain needed configuration parameters. There is also a Configuration Registry application that can be used to view and edit configuration data in the database. Changes to the Configuration Registry typically are under the control of Configuration Management and the Database Administrator.

Figure 10 shows the Configuration Registry application. Data in the Registry database are stored hierarchically, sorted by hostnames. An application may have multiple entries, each under a different hostname where it runs. The application queries the database directly. The application shows the Attribute Tree on the left. It displays parameters and their values in the Attribute Listing on the right. It provides the operator with capability to create, read, update, and delete database entries. A click on a parameter name in the Attribute Listing displays a pop-up window (see Figure 11), permitting the user to update or delete the parameter.

Use the following procedure to display configuration parameters.

Displaying Configuration Parameters

---

1 On workstation ***x*0dms##**, in a terminal window, type **/usr/ecs/*mode*/CUSTOM/utilities/EcCsRegistryGUIStart *mode* &** at a UNIX command prompt and then press the **Return/Enter** key (where ***mode*** is likely to be **TS1**, **TS2**, or **OPS**).

- The ***x*** in the workstation name is a letter designating your site: **g** = GSFC, **m** = SMC, **l** = LaRC, **e** = EDC, **n** = NSIDC, **o** = ORNL, **a** = ASF, **j** = JPL; the **##** is an identifying two-digit number (e.g., **g0dms03** indicates a data management subsystem workstation at GSFC). If you access the workstation through a secure shell remote login (ssh), you must enter **setenv DISPLAY <local_workstation IP address>:0.0** prior to the ssh before entering the command after the ssh. The **<*ipaddress*>** is the ip address of ***x*0mss##**, and **xterm** is required when entering this command on a Sun terminal.
- The Database Login window is displayed with entries filled in for **User Id:** (e.g., **EcCsRegistry**), **Server:** (e.g., ***x*0icg02_srvr**), and **DB Name:** (e.g., **EcCsRegistry_*mode***).

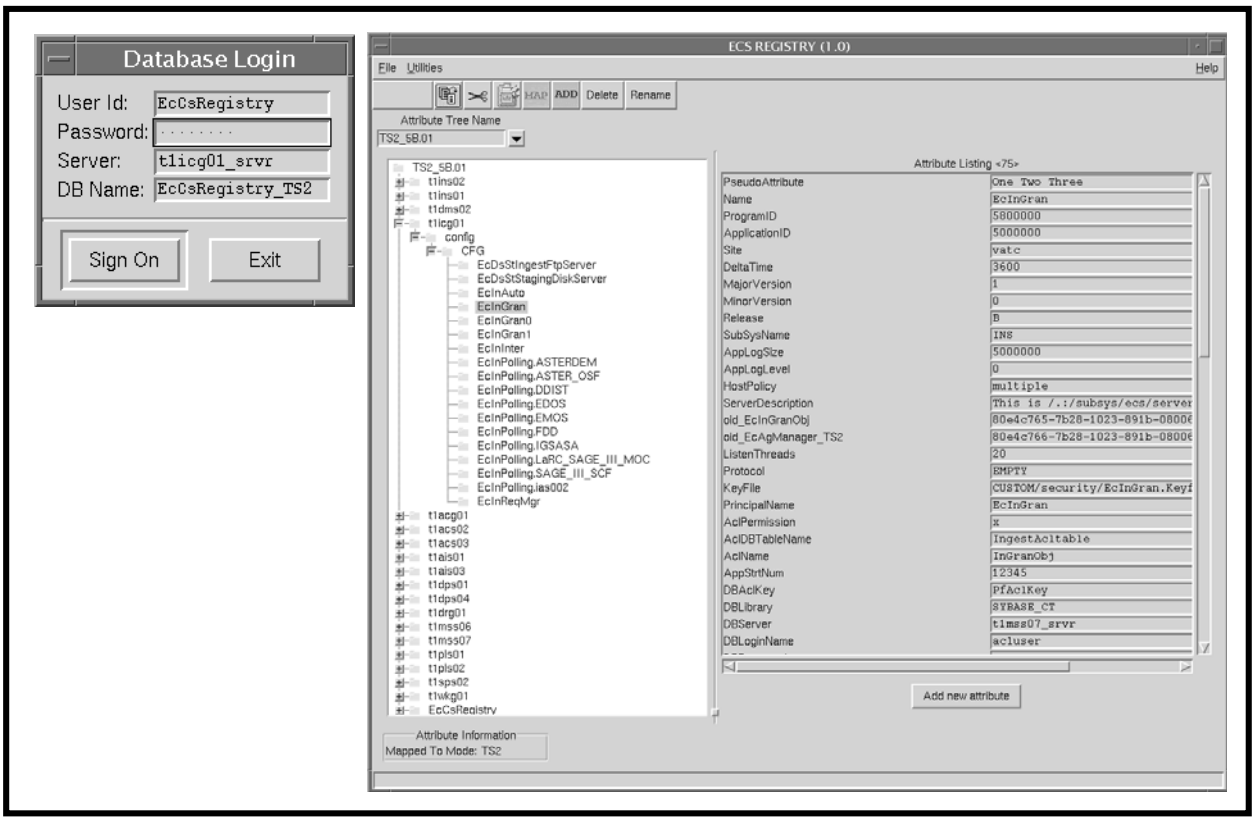

Figure 10. ECS Configuration Registry

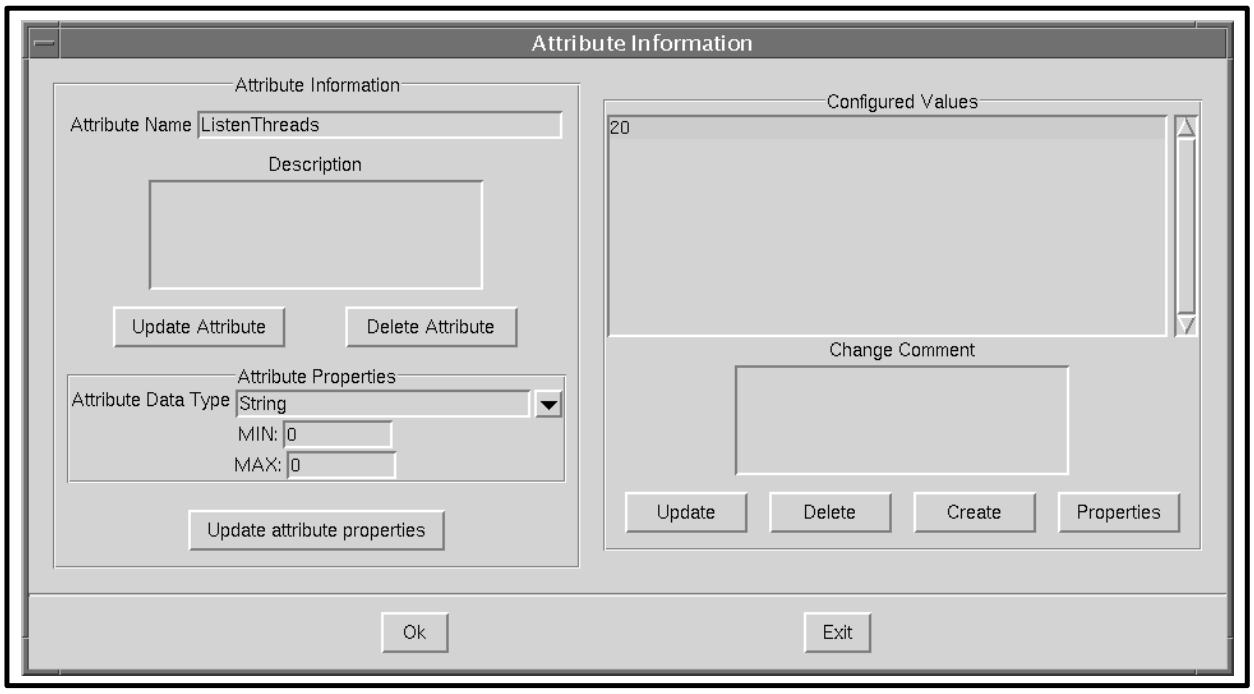

Figure 11. Configuration Registry Attributes Pop-Up Window

2 In the Database Login window, click in the **Password:** field and type the password.

- The typed password is not displayed (dots are displayed in place of the password).

3 Click on the **Sign On** button.

- The Database Login window is closed and the Configuration Registry GUI is displayed.

4 On the tree showing system hosts displayed on the left side of the GUI, click on the "+" sign next to one of the hosts for which parameters are to be displayed.

- The tree displays a **config** branch.

5 Click on the "+" next to **config**.

- The tree displays a **CFG** branch.

6 Click on the "+" next to **CFG**.

- The tree displays the computer software components for the selected host.

7 Click on one of the listed components (or its folder icon).

- The **Attribute Listing** field displays the configuration parameters associated with the selected component.

- If there are a large number of parameters, the right side of the window will have a scroll bar that may be used to scroll down the list.

8 Click on one of the listed parameters.

- The **Attribute Information** pop-up window for the selected parameter is displayed, showing detailed information concerning the parameter.
- If you are logged in with an account authorized with appropriate permissions, the **Attribute Information** window permits changing or deleting the parameter.

9 To exit from the Configuration Registry GUI, follow menu path **File→Exit**.

This page intentionally left blank.

Working with Indexes, Segments, and Caches

Indexes

An index provides a means of locating a row in a database table based on the value of a specific column(s), without having to scan all data in the table. When properly implemented, indexes can significantly decrease the time it takes to retrieve data, thereby increasing performance. Sybase allows the definition of two types of indexes:

- *Clustered index, where the rows in a database table are physically stored in sequence determined by the index.* The SQL Server continually sorts and re-sorts the rows of a table so that their physical order is always the same as their logical (or indexed) order. Clustered indexes are particularly useful when data are frequently retrieved in sequential order. Only one clustered index can be defined for a table.
- *Non-clustered indexes differ from their clustered counterpart in that the physical order of rows is not necessarily the same as their indexed order*. Each non-clustered index provides access to the data in its own sort order, giving the appearance of data in that physical order. Up to 249 non-clustered indexes can be defined for one table.

Clustered indexes allow faster searches than non-clustered indexes. Clustered indexes are often called the primary key of the table. If you don't specify which type of index you want, it will be a non-clustered index by default.

There are a lot of options for creating an index, but the most commonly used one is unique. Both clustered and non-clustered indexes can be unique. *A unique index prohibits duplicate values in the column that the index is on.* They cannot be created on text or image datatypes because those datatypes cannot reliably be declared unique.

Index keys need to be differentiated from logical keys. Logical keys are part of the database design, defining the relationships between tables: primary keys, foreign keys, and common keys. *When you optimize your queries by creating indexes, these logical keys may or may not be used as the physical keys for creating indexes.* You can create indexes on columns that are not logical keys, and you may have logical keys that are not used as index keys.

Tables that are read-only or read-mostly can be heavily indexed, as long as your database has enough space available. If there is little update activity, and high select activity, you should provide indexes for all of your frequent queries. Be sure to test the performance benefits of index covering.

Also, remember that with each insert, all non-clustered indexes have to be updated, so there is a performance price to pay. The leaf level has one entry per row, so you will have to change that row with every insert.

Deleting an Index

1 To delete an index, type **drop index**.

2 Type the table name that has the index, a period, and the index name that you want to delete.

3 Type **go** then press **Enter**.

Segments

Segments are named subsets of the database devices available to a particular SQL Server database. A segment can best be described as a label that points to one or more database devices.

Segmenting can increase SQL Server performance and give the SA or DBA increased control over placement, size and space usage of specific database objects. For example:

- If a table is placed on one device and its non-clustered indexes on a device on another disk controller, the time required to read or write to the disk can be reduced since disk head travel is usually reduced.
- If a large, heavily used table is split across devices on two separate disk controllers, read/write time may be improved.
- The SQL Server stores the data for text and image columns on a separate chain of data pages. By default, this text chain is placed on the same segment as the table. Since reading a text column requires a read operation for the text pointer in the base table and an additional read operation on the text page in the separate text chain, placing the text chain on a separate physical device can improve performance.

Segments are created within a particular database from the database devices already allocated to that database. Each SQL Server database can contain up to 32 segments. The database devices must first be initialized with **disk init** and then be made available to the database with a **create database** or **alter database** statement segment names can be assigned.

When a database is first created, the SQL Server creates three segments in the database:

- System, which stores the database's system tables.
- Logsegment, which stores this database's transaction log.
- Default, which initially stores all other database objects.

Additional segments can be created. ECS subsystem databases, for example, consist of the following:

- Default data segment used if no other segment specified in the create statement.

- SYSLOGS, transaction logs.
- System tables and indexes.
- OPS mode data segment.
- OPS mode index segment.
- TS1 mode data segment.
- TS1 mode index segment.
- TS2 mode data segment.
- TS2 mode index segment.

If tables and indexes are placed on specific segments, those database objects cannot grow beyond the space available to the devices represented by those segments, and other objects cannot contend for space with them. Segments can be extended to include additional devices as needed. Thresholds can be used to provide warnings when space becomes low on a particular database segment.

In the following example procedure, the DBA receives a request to create a segment for the storage of the SubServer table indexes in the t1ins01_srvr database on a separate physical disk. Two devices, **subserver_data** and **subserver_index**, have already been created and are located on different physical disks.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database
- Database device extents
- Space on the database allocated to the Database device

To create a database, the DBA must have **sa_role**.

Creating Database Segments (Example)

---

1 At the UNIX prompt, type **cd scripts** and then press **Return**.

2 Using the text editor of your choice, edit ***segment*.sql** and make changes to information enclosed in brackets (including the brackets) as appropriate.

- Note: This file doesn't exist. It must be created.

3 After the changes have been made, save the file according to the rules of the text editor.

4 At the UNIX prompt, type:**isql –U<username> -S<*ServerName*> -i*Segment*.sql -o*Segment*.out** and then press **Return.**

- The system will prompt you for a password.

5 At the **Password:** prompt, type the <***password***>, then press **Return.**

- When the UNIX prompt is again displayed the process is complete.

6 At the UNIX prompt, type **more *Segment.out*** , then press **Return**.

- This allows you to view the ***Segment.out*** file to confirm that the device has been created or to check for database creation errors.

Figure 12 shows an example of creation of a database segment template file.

```
/*****************************************/
/* name: [segment.sql]                   */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/

 sp_addsegment [seg_name], [DBname],[Device Name]
gogo[DBDevice]
```

Figure 12. Example of Creation of a Database Segment Template File

After the segment has been defined in the current database, the **create table** or **create index** commands use the optional clause **"on segment_name"** to place the object on a particular segment. Syntax:

create table table_name (column_name datatype ...) [on segment_name]

create [clustered | nonclustered] index index_name on table_name (columns)

Use **sp_helpdb** database_name to display the segments defined for that database.

Use **sp_helpsegment** segment_name to list the objects on the segment and show the mapped devices.

Caches

A cache is a block of memory that is used by Sybase to retain and manage pages that are currently being processed. By default, each database contains three caches:

- Data caches retain most recently accessed data and index pages
- Procedure caches retain most recently accessed stored procedure pages
- User transaction log caches are transaction log pages that have not yet been written to disk for each user

The size of each of these default caches is a configurable item that must be managed on a DAAC-by-DAAC basis. These caches can be increased or decreased as needed. The data cache can be further subdivided into named caches. *A named cache is a block of memory that is named and used by the database management system (DBMS) to store data pages. The named data caches can be used only by databases or database objects that are explicitly bound to them. All objects not explicitly bound to named data caches use the default data cache.* A database, table, index, or text or image page chain can be bound to a named data cache.

Assigning a database table to named cache causes accessed pages to be loaded into memory and retained. Named caches greatly increase performance by eliminating the time associated for disk input and output (I/O).

This page intentionally left blank.

Backing Up and Recovering Data

Database and transaction log backups and recoveries are probably the most important tasks the DBA must perform. Without regular and complete backups of databases and transaction logs, the potential for enormous amounts of data to be lost is very great. How often you back up a database depends on how frequently the data is updated and how costly it would be to lose it. Databases that are constantly being modified need to be backed up frequently; databases containing relatively static data can be backed up less frequently. You should regularly back up production databases, for example. If you experience a media failure, you can restore the data from the most recent backups the **restore** command. You do not have to shut down servers to back up or restore a database. The Backup Server makes it possible to perform online backup and restore operations while the Adaptive Server is running.

Backups

Each DAAC is responsible for determining its own backup schedule to meet its individual requirements. It is strongly suggested that a regular schedule of database backups be established and maintained. Although the databases are included in the daily backups of the entire system, separate database backups should be performed nightly.

Database backups are run by a UNIX **cron** job which executes the **sybasedump** program located in the **$SYBASE** directory. No intervention in the automatic backup process is required by the DBA, though periodic checks of the backup subdirectories are recommended.

Manual backups can be performed at any time by the DBA and are recommended for the following situations:

- Any change to the master database, including new logins, devices, and databases.
- Any major change to user databases, such as a large ingest or deletion of data, definition of indexes, etc.
- Other mission-critical activities as defined by the DAAC operations controller.

Automatic Backups

The following are procedures and scripts files that are currently being used for Sybase backups. There are **cron** jobs running at all Sybase servers that have SQL server installed. All dump files are currently written to local machine. The site DBA is responsible for configuring the backup dump to the remote Sybase directory.

1 Check if **crontab** is up and running.

> **crontab -l**

- The output will look something like the following:

  019 * * 1-6
  /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpDb
  012 * * 1-6
  /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpTran
  021 * * 1-5
  /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_CkErrorLog

2 If **crontab** is not running, enter:

> **crontab /usr/ecs/OPS/COTS/sybase/run_sybcron**

- The files shown in Table 10 will be installed by ECS Assistant to the directory **/usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin**:

Table 10. Automatic Backup Files

Script	Description
EcCoDbSyb_SetupKsh	This file contains the SYBASE and DSQUERY (server) environment setup. This file is called by EcCoDbSyb_DumpDb, EcCoDbSyb_DrumpTran, and EcCoDbSyb_CkErrorLog scripts.
EcCoDbSyb_DumpDb	This script contains the code to dump the databases. First, it checks for any DBCC error on the master database, if there is any error on the master, the script sends an email to the DBA and exits the program. If the master database dump was successful, then the rest of the databases are dumped. Each database has a DBCC check; if there is any error on the database then the database is NOT dumped and an email is sent to the DBA. At the end, a status email is sent, providing all the database names that were successfully dumped.
EcCoDbSyb_DumpTran	This script contains the code to dump the transaction logs. This dumps the transaction logs for each database, it checks the error log file; if the error Msg is 4207 or 4221 it does a dump of the database first, then it does the transaction dump. If there is any other error Msg then the transaction dump fails and email is sent. At the end, the status of the transaction log dumps is emailed to the DBA.
EcCoDbSyb_SedFile	This file contains all the database that don't need to be dump (i.e., temp, model, etc.)
EcCoDbSyb_DboMail	This file contains the email list of all the DBAs.

Table 10. Automatic Backup Files

Script	Description
EcCoDbSyb_DbStat	This script updates the index table of a database. This script is called from EcCoDbSyb_DumpDb after each successfully database dump.
EcCoDbSyb_CkErrorLog	This script checks for specific database error messages from the Sybase Error Log File every hour and emails the error messages to the DBAs in the EcCoDbSyb_DboMailfile.
EcCoDbSyb_tran_log.awk	This script matches the current hour with the hour the error messages were generated in the Error Log File. If errors found, the messages are saved in a mail file and sent to DBAs.
0 0 * * 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpDb 0 10,13,16 * * 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpTran 0 * ** 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_CkErrorLog NOTE: Make sure there is an OPS mode directory with all script files.	

3 Modify the files shown in Table 11 before running any of the scripts shown in Table 10.

Table 11. Files That Need to Be Modified before Running Scripts

File	Modification
EcCoDbSyb_SetupKsh	Make user you have the SYBASE files under /usr/ecs/OPS/COTS/sybase
EcCoDbSyb_SedFile	Add any other database that might not need to be backed up. The databases that are listed in this file do not need to be backed up.
EcCoDbSyb_DboMail	Add/delete the email of the DBA and any other email that might need to be added/deleted. All the errors and status are sent to them.

SQL Server backups are performed nightly by a **cron** job which runs the **run_sybcron** program located in the **$SYBASE**/ directory. The following table of definitions (Table 12) is used throughout the rest of this section.

Table 12. Automatic Backup Components

Name	Function
run_sybcron	File added with the crontab -e command, contains several executable cron commands. **Example:** 00 19 * * 1-6 /data1/COTS/sybase/sybasedump
EcCoDbSyb_DumpDb	Controlling script that performs the following functions: run isql to create the Backup Statements run isql to execute the Backup Statements record the results of the Backup Statements in Backup Files copy the Backup Files to the Backup Subdirectory create the Backup Summary greps successful Dump statements along with any errors generated, sends e-mail to the DBA and writes them to the backup_summary file
sp	SQL Server password file - contains password for backup role

Manual Backups

Both the **dump database** and **dump transaction** command processing are off-loaded to the backup server and will not affect normal operations of the database. These commands are performed by the System Administrator on appropriate databases using the following syntax.

1> dump database master to
"/usr/ecs/OPS/COTS/sybase/sybase_dumps/dumps/dbname.dat.MMDDHHMM."

go

After dumping the database, compress the dump file by executing:

%compress
/usr/ecs/OPS/COTS/sybase/sybase_dumps/dumps/dbname.dat.MMDDHHMM

Perform Manual Backups

---

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server that contains the database to be backed up by typing: **/tools/bin/ssh *gsfcsparc5.gsfcmo.ecs.nasa.gov***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key.

- Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into Sybase by typing: **su-sybase**, then press **Return.**

- A password prompt is displayed.

7 Enter ***SybasePassword*** then press **Return**.

- Remember that ***SybasePassword*** is case-sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

8 At the UNIX prompt, type **isql -Usa** then press **Return**.

9 To backup the database, at the ISQL prompt, type **dump database *DBName* to *BackupDirectory/DBName*.dat**, then press **Return**.

10 Type **go**, then press **Return**.

11 To backup the transaction log, at the ISQL prompt, type **dump transaction *DBName* to *BackupDirectory/DBName*_tx.dat**, then press **Return**.

12 Type **go**, then press **Return**.

Database Recovery

Database recovery is performed when the Database Administrator determines that some database data is corrupt or when the System Administrator determines that a database device has failed. The System Administrator makes a request to the Database Administrator, who performs the restore and notifies the System Administrator when the restore is complete.

The symptoms of media failure are as variable as the causes. If only a single block on the disk is bad, your database may appear to function perfectly for some time after the corruption appears; this is why you should run **dbcc** commands frequently. If an entire disk or disk controller is bad, you will not be able to use a database. SQL Server marks it as suspect and displays a warning message. If the disk storing the master database fails, users will not be able to log into the server, and users already logged in will not be able to perform any actions that access the system tables in master.

The complete database device restoration procedure is actually a suite of procedures that must be performed in the prescribed order:

1 The device failure has been verified by the System Administrator and restoration has been requested from the DBA.

2 If possible, perform a dump of the current database and the current database transaction log for each database on the failed device to be restored. If this is not possible due to the damage to the database device, then the DBA will have to use the most recent database and transaction log dumps.

3 If possible, the DBA examines the space usage for each database on the failed device. The same defaults should be set when the new database device(s) is (are) created.

4 The DBA drops the database(s) on the failed device, then the database device itself is dropped.

5 The DBA initializes a new database device. This is why it is important to keep the device creation scripts.

6 The DBA recreates each user database on the new database device. This is why it is important to keep the user creation scripts.

7 Each database is reloaded with data from the database backups and the transaction log backups. It is this procedure that is detailed below.

8 The DBA notifies the System Administrator when the database restoration is complete.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device that has failed
- Name of the replacement device
- Name of the databases that reside on the failed device
- Name of the backup volumes
- Name of the dump files on the backup volumes

Manual Recovery

Manual recovery of a user database is performed by the System Administrator by the use of the **load database** and **load transaction** commands. For issues concerning the **master** database, please consult your System Administrator's Guide for assistance. It is recommended that any user database to be recovered be dropped and created with the **for load** option., The **databasename**.sql along with any **alter.databasename**.sql scripts can be combined into one script which will re-create the user database with the **for load** option. This will insure the success of the **load database** and **load transaction** commands.

In the example procedures that follow, the database **UserDB** was created using the following script excerpt (stored in home/scripts/create.databases/userdb.sql):

create database UserDB on data_dev1 = 100 log on tx_log1 = 50 [with override]

and was modified using the following script excerpt (home/scripts/create.databases/alteruserdb.sql):

alter database UserDB on data_dev1=50

For the purposes of this example, the full database backup and transaction log dumps were successful and located in **/usr/ecs/OPS/COTS/UserDB.dat** and **UserDB_tx.dat**.

Perform a User Database Recovery (Example)

---

1 In the $SYBASE/scripts/create.databases directory, make a script file from the template.

- Syntax:

  % cd /usr/ecs/OPS/COTS/sybase/scripts/create.databases, % cp template.sql userdb_for_load.sql.

2 Modify appropriate items so that the script file resembles the following:

1> create database UserDB on data_dev2=100 log on tx_log2=50 for load
2> go
3> alter database UserDB on data_dev3=50
4> go

3 Save the script in **$SYBASE/scripts/create.databases/userdb_for_load.sql**

4 Run the script from the UNIX command prompt.

%isql -Usa -Sservername -iuserdb_for_load.sql -ouserdb_for_load.out

5 Check the **userdb_for_load.out** file for success.

6 Load the database from the full backup.

1> load database UserDB from
"/usr/ecs/OPS/COTS/sybase/sybase_dumps/week1/dbname.dat.MMDDHHMM"
go

7 Load the transaction file from the transaction file dump.

1> load transaction UserDB
from"/usr/ecs/OPS/COTS/sybase/sybase_dumps/week1/dbname.tran.MMDDHHMM"
3> go

---

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server on which the new database device is to be created by typing: **/tools/bin/ssh <*ServerName*>**, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

NOTE: For each database device to be created, perform Steps 6 through 9:

6 At the UNIX prompt, type **cd server.databases** then press **Return.**

- This places you in the directory that contains all the scripts used to create database devices.
- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see **Restore Databases**).

7 At the UNIX prompt, type **cp template.sql *DBName*.sql** then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the user database.

8 Using the text editor of your choice, edit ***DBName*.sql** and make changes to information enclosed in brackets (including the brackets) as appropriate (see Figure 13).

- **DBName** is the name of the database and should describe its function succinctly.
- **DBDeviceName** is the name of the existing, approved database device on which **DBName** will reside.
- **DBSize_in_MB** is the estimated size of the database in megabytes.
- **LogDBDeviceName** is the name of the database device holding the transaction log
- **LogSize_in_MB** is the estimated size of the transaction log in megabytes.

- The **sp_helpdb** command provides feedback at the end of the script to assure that the database has been created.

```
/* ********************************************** */
/* name: template.sql                             */
/* purpose:                                       */
/* written:                                       */
/* revised:                                       */
/* reason:                                        */
/* ********************************************** */
create database database_name
on data_dev = size, /* in MB */
log on log_dev = size
go
sp_helpdb database_name
go
```

Figure 13. Sample template.sql File for Creation of a Database

9 After the changes have been made, save the file according to the rules of the text editor.

10 At the UNIX prompt, type:

isql –U<username> -S<ServerName> -i*DBName*.sql -o*DBName*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***DBName.sql*** is the name of the script file you created in step 8.
- ***DBName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes. The system will prompt you for a password.

11 At the **Password:** prompt, type the **<*password*>**, then press **Return.**

- When the UNIX prompt is again displayed the process is complete.

12 At the UNIX prompt, type **more *DBName.out*** then press **Return**.

- This allows you to view the ***DBName.out*** file to confirm that the device has been created or to check for database creation errors.

A sample of a completed **Create Database** script is shown in Figure 14.

```
/*****************************************/
/* name: test_db.sql                     */
/* purpose: create a database for testing */
/* written: 12/19/97                     */
/* revised:                              */
/* reason:                               */
/*****************************************/
create database test_db
on test_dev = 3
go
sp_helpdb test_db
go
```

Figure 14. Completed Create Database Script

Establishing Database Security

Discretionary Access Controls

Permissions control access within a database. There are two types of permissions within a database: object and command. Object privileges control select, insert, update, delete, and execute permissions on tables, views, and stored procedures. Command permissions control access to the create statements for databases, defaults, procedures, rules, tables, and views. Access controls allow giving various kinds of permissions to users, groups, and roles with the **grant** command. The **revoke** command rescinds these permissions.

The ability to assign permissions for the commands is determined by each user's status (e.g., system administrator, database owner, database object owner) and by whether or not a particular user has been granted a permission with the option to grant that permission to other users.

Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group.

Roles

Roles provide a structured means for granting users the permissions needed to perform standard database-related activities and also provide a means for easily identifying such users. They provide a means of enforcing and maintaining individual accountability. There are six pre-defined roles shown in Table 13 that may be assigned to a user. Other unique roles can be created and granted to users, groups of users, or other roles. In addition, role hierarchies can be defined. This enables a user assigned one role to automatically have the privileges available to all other roles lower in the hierarchy. Mutually exclusive roles can also be defined. This can be done in terms of membership – a single user cannot be granted both roles – or activation – a single user can be granted two roles but cannot enable both (e.g., both cannot be enabled simultaneously).

Table 13. Roles and Privileges

Roles		Privileges
System Administrator	sa_role	Grant a specific user permissions needed to perform standard system administrator duties including: Installing SQL server and specific SQL server modules Managing the allocation of physical storage Tuning configuration parameters Creating databases
Site Security Officer	sso_role	Grant a specific user the permissions needed to maintain SQL server security including: • Adding server logins • Administrating passwords • Managing the audit system • Granting users all roles except the sa_role
Operator	oper_role	Grant a specific user the permissions needed to perform standard functions for the database including: • Dumping transactions and databases • Loading transactions and databases
Navigator	navigator_role	Grant a specific user the permissions needed to manage the navigation server
Replication	replication_role	Grant a specific user the permissions needed to manage the replication server
Sybase Technical Support	sybase_ts_role	Grant a specific user the permissions needed to execute database consistency checker (dbcc), a Sybase supplied utility supporting commands that are normally outside of the realm of routine system administrator activities

Granting or Revoking Database Access Privileges

The DBA uses the **grant** and **revoke** statements to control permissions. The syntax for grant and revoke statements is similar:

grant {all [privileges] | command_list }to { public | name_list | role_name }

revoke {all [privileges] | command_list } from { public | name_list | role_name}.

Granting or Revoking Database Access Privileges

1 Determine the privileges that you want to grant and to which user(s).

2 Log on to a local host.

3 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

4 Log into the server where the user database resides by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 5.
- If you have not previously set up a secure shell passphrase; go to Step 6.

5 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key.

- Go to step 7.

6 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

7 At the UNIX prompt, type **su-sybase** then press **Return**.

- A password prompt is displayed.

8 Enter ***SybasePassword***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

9 At the UNIX prompt, type **isql -Usa**, then press **Return**.

10 At the **Password:** prompt, type the ***SybaseSAPassword*** then press **Return.**

- Note: the ***SybaseSAPassword*** is case-sensitive.

11 If you are granting privileges, at the ISQL prompt, type **grant *Privilege* to *LoginName*** then press **Return**.

- If more than one privilege is to be granted, repeat this step on subsequent lines.

12 If you are revoking privileges, at the ISQL prompt, type **revoke *Privilege* from *Logname*** then press **Return**.

- If more than one privilege is to be revoked, repeat this step on subsequent lines.

13 When all privileges have been assigned, type **go**, then press **Return**.

NOTE: It is recommended that the DBA store the privilege granting and revoking commands in **.sql** files in the **$SYBASE/scripts/..** directory along with their results.

Identification and Authentication Controls

A user is given a login account with a unique ID. All of that user's activity on a server can be attributed to his or her server user ID and audited. Passwords are stored in the **master..syslogins** table in encrypted form. When users log in from a client, they can choose client-side password encryption before sending the password over the network. A user can be granted the ability to impersonate another user. This proxy authorization allows administrators to check permissions for a particular user or to perform maintenance on a user's database objects. Application servers can execute procedures and commands on behalf of several users.

Providing users with access to SQL servers and their databases consists of the following steps:

- A server login account for a new user is created.
- The user is added to a database and optionally assigned to a group.
- The user or group is granted permissions on specific commands and database objects.

The user needs a SQL Server login to access the ECS DBMS. The login is assigned to a user with the related permissions. During initial database installation, logins used by the ECS custom code were created and permissions assigned for access to subsystem databases. In addition, special database installation logins, **<SUBSYS>_role**, were created to support database installation needs. For each login, the level of access is limited to that associated with their login, group or assigned group/role. Object permissions are set within the installation scripts of the <SUBSYS> subsystem for each object and group/role.

In order for a user to connect to a SQL Server, a login must be created by the DBA. That login must then be assigned to particular database(s) that the user will access. All login details are stored in the syslogins table in the **master** database. Two stored procedures are required:

- The stored procedure **sp_addlogin** adds new login names to the server but does not grant access to any user database. Syntax:

  sp_addlogin login_name, password, [,default database ,language, fullname]

- The stored procedure **sp_adduser** adds the user. Syntax:

  1> **sp_adduser**login_name [username, group_name]
  2> **go**

Create an SQL Server Login

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server for which a new login procedure is to be created up by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 At the UNIX prompt, type **su-sybase** then press **Return**. A password prompt is displayed.

7 Enter ***SybasePassword*** then press **Return**.

- Note: ***SybasePassword*** is case-sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is /usr/ecs/OPS/COTS/sybase and all required environment variables have been set.

8 At the UNIX prompt, type **cd scripts/server.users** then press **Return**.

9 At the UNIX prompt, type **cp template.sql *UserName*.sql** then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the new user login.

10 Using the text editor of your choice, edit ***UserName*.sql** (see Figure 15) and make changes to information enclosed in brackets (including the brackets) as appropriate:

11 After the changes have been made, save the file according to the rules of the text editor.

12 At the UNIX prompt, type **isql -Usa -S*ServerName* -i*UserName*.sql -o*UserName*.out** then press **Return**.

- ***ServerName*** is the name of the server on which the master database is stored.
- ***UserName*.sql** is the name of the script file you created in Step 9.
- ***UserName*.out** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

```
/*****************************************/
/* name: [template.sql]                  */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/
     sp_addlogin [login name], [password], [default database]
     go
     use [default database]
     go
     sp_adduser [login name]
     go
     /* the following is optional, by database */
     sp_changegroup [group name], [login name]
     go
     sp_helpuser
go
```

Figure 15. Sample template.sql File for New Database User Login

13 At the Password: prompt, type the ***SybaseSAPassword***, then press **Return**.

- When the UNIX prompt is again displayed the process is complete.

14 At the UNIX prompt, type **more *UserName*.out** then press **Return**.

- This allows you to view the ***UserName*.out** file to confirm that the user login has been created or to check for user login creation errors.

One of the most common problems a DBA encounters is a user who cannot connect to a SQL Server. The following procedure is applicable.

Changing a Password

---

1 At the UNIX prompt, type **isql -U*dbastudent*** then press **Return.**

- The system will prompt you for a password.

2 At the **Password:** prompt, type the ***dbastudentpassword*** then press **Return.**

3 At the Sybase prompt, type the following:

sp_password ***old-password, new-password***

NOTE: The dba (or any user with sso_role) may change another user's password with the following syntax: (this is the most common activity performed)

sp_password ***sso_role password, new-password, login name***

Auditing

A comprehensive audit system is provided with Adaptive Server. The audit system consists of a system database called sybsecurity, configuration parameters for managing auditing, a system procedure, **sp_audit**, to set all auditing options, and a system procedure, sp_addauditrecord, to add user-defined records to the audit trail. When you install auditing, you can specify the number of audit tables that Adaptive Server will use for the audit trail. If you use two or more audit tables to store the audit trail, you can set up a smoothly running audit system with no manual intervention and no loss of records.

A System Security Officer manages the audit system and is the only user who can start and stop auditing, set up auditing options, and process the audit data. As a System Security Officer, you can establish auditing for events such as:

- Server-wide, security-relevant events.
- Creating, deleting, and modifying database objects.
- All actions by a particular user or all actions by users with a particular role active.
- Granting or revoking database access.
- Importing or exporting data.
- Logins and logouts.

The following statements represent examples for performing auditing:

- Run sybinit and install auditing.
- Add a login for auditing:

  sp_addlogin ssa, ssa_password, sybsecurity
  use sybsecurity
  sp_changedbowner ssa
  sp_role "grant", sso_role, ssa

- Enable auditing:

  sp_auditoption "enable auditing", "on"
  sp_auditlogin loginname, "cmdtext", "on"

- To Test:
  - Create a table in a database with one field.
  - Grant all on the table for the **loginname.**
  - Log into isql using the **loginname.**
  - Insert a record into the table.
  - Log into isql as **ssa.**
  - Select * from sysaudits where **loginname** = **“loginname”**

NOTE: Once auditing is turned on, the **sysaudit** tables will get filled up very quickly. The database option options of **trunc log on chkpt** needs to be turned on in the **sybsecurity** database so that the **logsegment** of this database can be cleaned up upon checkpoint. This **logsegment** can also be cleaned up by means of transaction dumps if the database is installed on two separate disk devices. Also, as soon as auditing is turned on, a **cron** job needs to be turned on at the same time to **bcp** out and truncate the **sysaudit** table. This is very important.

ECS Security Directive

All System Administrators and Database Administrators at the sites are responsible for reasonable security measures when installing ECS custom software. This means:

- Changing the permissions of online secure files to the minimum level required.
- Backing up secure file(s) to removable media (floppy or tape) and removal of secure files immediately after installation is complete and then keeping the removable medium in a secure location.

The following file is affected as result of this requirement on the ECS program:

/usr/ecs/<MODE>/CUSTOM/dbms/<SUBSYSTEM>/Ec<server>SybaseLogins.sql

Set permissions to 711 (user read, write, execute, group and other read only).

Copying, Replicating, and Extracting Data

Copying Databases

Individual Databases

To create an exact copy of a database:

- Dump the existing database.
- Create a database to load with this dump.

The new database does not have to be the same size as the original. The only requirement is that the destination database must be at least as large as the dumped database and have the same beginning fragments as the original database. This information can be obtained from saved database creation scripts or by running the following command:

select segmap,'Size in MB'=size/512 from sysusages where dbid= db_id("database_name")

Copying a Database (Example)

A database was created with the following statement:

```
create database dbname on datadevice1 = 1000,
log on Logdevice1 = 200
go
alter device dbname on datadevice2 = 500     running:
select segmap,'Size in MB'=size/512 from sysusages
where dbid= db_id("dbname")
would return:segmap  Size in MB
3       1000
4       200
3       500
```

You could create a 3GB database as follows and load your database into it (using **for load** option will shorten database load time):

```
create database newdatabase on datadevice3 = 1000 log on logdevice3 = 200
for load
go
alter database newdatabase on datadevice 3=500 for load go
alter database newdatabase on datadevice4=300 for load go
alter database newdatabase on datadevice5=1000 for load go
load database newdatabase from dbname_dump go
```

Bulk Copying

The bulk copy (**bcp)** utility is located in the **$SYBASE/bin** directory and is designed to copy data to and from SQL Server databases to operating system files. You must supply the following information for transferring data to and from SQL Server:

- Name of the database and table.
- Name of the operating system file.
- Direction of the transfer (in or out).

In order to use **bcp**, you must have a SQL Server account and the appropriate permissions on the database tables and operating system files that you will use. To copy data **into** a table, you must have **insert** permission on that table. To copy data **out** to an operating system file, you must have **select** permission on the following tables:

- The table being copied:
  - **sysobjects**
  - **syscolumns**
  - **sysindexes**

bcp syntax:

bcp [[database_name].owner.]table_name {in | out} datafile [-e errfile] [-n] [-c] [-t field_terminator] [-r row_terminator] [-U username] [-S server]

Perform Bulk Copying

---

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server that contains the database to be backed up by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase, go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key.

- Go to step 6.

5 At the ***<user@remotehost>*'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Set Login as sybase and cd to /usr/ecs/OPS/COTS/sybase/scripts/server.bcp.

- Note: ***SybasePassword*** is case-sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

7 Copy the bcp_script template to [table name]_out.

8 Modify the script with the correct database name, table name, direction, and login name.

9 Run the [table name]_out script and press **Return** for each of the prompts except the last.

10 At the last prompt, store [your responses] in a [table name].fmt file. This creates a format file for future bulkcopy activity.

11 To copy the data to another already created table, repeat Steps 6, 7, and 8 with the following changes:

- the direction is “in”
- use the optional **-f [format file name]**

This page intentionally left blank.

Replication System Administration

Administering the replication system is primary the role of the Replication System Administrator (RSA). The Replication System Administrator installs, configures, and administers the replication system. Given the distributed nature of the MSS implementation this role may be performed by different people at different locations. If this is the case, various tasks for administering Replication Server may require coordination between Replication System Administrators.

The Replication System Administrator has **sa** user permissions, which provides that person with the ability to execute nearly all commands in the replication system. In managing the system, the Replication System Administrator may need to coordinate with DBAs for both local and remote databases.

Replication System Administrators should be experienced Sybase DBAs and should have taken the Sybase training classes *Replication System Administration* and *Replication Disaster Recovery Workshop*. They should also have read and understood the manuals: *Replication Server Administration Guide, Replication Server Configuration Guide for UNIX Platforms, Replication Server Reference Manual,* and *Replication Trouble Shooting Guide*.

Replication System Administrator Tasks

Table 14 identifies tasks required to maintain a replication system:

Table 14. Replication System Administrator Tasks

Task	Roles
Installing Replication Server	Replication System Administrator (RSA)
Adding or removing a Replication Server	RSA
Starting up and shutting down Replication Server.	RSA
Configuring Replication Server	RSA
Maintaining Routes (Creating and modifying)	RSA
Managing the RSSD	RSA
Adding a primary and replicate database.	RSA
Adding login names, database users, and administering appropriate permissions	RSA
Adding replicated tables or changing table schemas. Creating and modifying replicated tables Creating and modifying replication definitions Creating and materializing subscriptions at replicate sites.	RSA
Defining data server function-string classes and function strings.	RSA
Applying database recovery procedures.	RSA

Table 14. Replication System Administrator Tasks

Task	Roles
Maintaining and monitoring database connections	RSA
Monitoring Replication Server	RSA
Processing rejected transactions	RSA
Quiescing Replication Server	RSA
Reconciling database inconsistencies.	RSA

DAAC DBA Replication Roles and Tasks

The Database Administrator (DBA) supports some Replication Server administrator tasks. Table 15 lists tasks that the DBA administrators perform at the local DAACs with respect to replication server administration.

Table 15. DAAC DBA Replication Roles and Tasks

Task	Roles
Installing Replication Server	DBA
Managing the RSSD	DBA
Adding a primary and replicate database.	DBA
Adding login names, database users, and administering appropriate permissions	DBA
Adding replicated tables or changing table schemas. Creating and modifying replicated tables Creating and modifying replication definitions Creating and materializing subscriptions at replicate sites.	DBA
Defining data server function-string classes and function strings.	DBA
Applying database recovery procedures.	DBA
Processing rejected transactions	DBA
Quiescing Replication Server	DBA
Reconciling database inconsistencies.	DBA

Database Replication

The MSS database is the only database to use replication. This is done so user profile and order tracking data can be shared among the DAACs.

Replication begins with definitions and subscriptions. *Replication definitions specify the tables and columns that may be replicated for a data recipient. Replications subscriptions specify which definitions the data recipient wishes to receive*. In addition, the replication database and

server must be configured to support the potentially large volumes of data that will be transferred between the source and recipient databases.

The MSS replication definition and subscription scripts were developed as templates for installation at DAACs and the SMC. Since peer-to-peer configuration for MsAcUsrProfile is required, the template approach was decided so that only those scripts that need to be implemented at each site are configured. The template files provide the necessary generic functions needed to configure the MSS MsAcUsrProfile replication environment. Unix shell scripts have been developed to allow installers to pass the appropriate site-specific information (i.e. database name, replicate replication server name, etc.) when prompted. The Unix script then customizes the template script files for the specific site. The DD&M group has created naming conventions for all replication scripts and naming conventions for the subscriptions, replication definitions, and other objects related to the ECS replication environment. Script names consist of the following conventions:

<action>.<SUBSYS>.<Replication Object>.<Primary Site ID>.sql.<*Replicate Site ID*>

This page intentionally left blank.

Performance Monitoring, Tuning, and Problem Reporting

Databases are ubiquitous in ECS, and performance has been addressed from the beginning (see *Plan for Achieving Required ECS Throughput Performance*, 241-TP-002-001). Each of these databases has an expected performance profile, including initial size, growth rates, transaction rates, and response times. Analysis of database performance generally takes a multi-tiered approach. At the lowest level, the data access patterns are analyzed and the database physical design is tailored for these access patterns. These activities occur during the design stage of the development. During the implementation phase, actual transactions are inspected to make sure that they are efficiently coded, and are consistent with the expected use of the database. Once implemented, the performance of the database is measured under various loads; performance is tuned by modifying various configuration parameters provided by the database management system, and in some cases by modifying the database design or the data access code.

Actual performance can be assessed from a response time perspective or a capacity perspective. Response times can be measured in a variety of ways. For some of the system databases, a direct user interface is available that will provide a basis for assessing response time performance—for example, searches provided through the client. For these cases, the analytical task becomes the separation of the database component of the response time from all other contributors. Other system databases do not have a direct user interface because they serve system internal functions; an example is the Subscription database. Response time performance for these databases may be measured using test drivers r by capturing timing marks from instrumented code as test threads are run through the system. Database capacity—usually measured as the number of transactions that the database can handle per second—is usually tested with test drivers, although it can also be measured when the system is put through thread tests, operational tests, or stress tests. Sybase monitoring tools can be used to measure the throughput of the system during such tests, although they add their own overhead and reduce the maximum capacity achievable. The Sybase monitoring tool collects an enormous wealth of information about the use of Sybase resources—everything from hardware (CPUs, memory, disk channels) to Sybase internally configured resources (cache blocks, spin locks, indexes). These resources can then be tuned according to the usage patterns observed (including the possible addition of hardware).

Monitoring

The easiest way to monitor performance is through the Monitor Server. The Monitor Server provides specific performance data on cache usage, network traffic, device I/O, and locking activity. *Performance is the measure of efficiency of an application or multiple applications running in the same environment*. Performance is usually measured in one of two ways:

- *Response time, which is the time that a single task takes to complete.*

- Throughput, which is the volume of work completed in a fixed time period, e.g. transactions per second (tps).

Most of your tuning efforts should address the amount of time it takes for the server to respond to queries.

You can run **sp_sysmon** both before and after tuning Adaptive Server configuration parameters to gather data for comparison. You can also use **sp_sysmon** when the system exhibits the behavior you want to investigate. In many tests, it is best to start the applications, and then start **sp_sysmon** when the caches have had a chance to reach a steady state. If you are trying to measure capacity, be sure that the amount of work you give the server keeps it busy for the duration of the test. Many of the statistics, especially those that measure data per second, can look extremely low if the server is idle during part of the sample interval. In general, **sp_sysmon** produces valuable information when you use it:

- Before and after cache or pool configuration changes.
- Before and after certain sp_configure changes.
- Before and after the addition of new queries to your application mix.
- Before and after an increase or decrease in the number of Adaptive Server engines.
- When adding new disk devices and assigning objects to them.
- During peak periods, to look for contention.
- During stress tests to evaluate an Adaptive Server configuration for a maximum expected application load.
- When performance seems slow or the system behaves abnormally.

Tuning

Response time can be shortened by reducing contention and waits times, particularly disk I/O wait times; using faster components; and reducing the amount of time the resources are needed. In some cases, Adaptive Server is optimized to reduce initial response time, that is, the time it takes to return the first row to the user. This is especially useful in applications where a user may retrieve several rows with a query and then browse through them slowly with a front-end tool. Other ways of increasing performance are summarized by system level in Table 16.

Table 16. Tuning Options

Layers	Tuning Options
Application	Remote or replicated processing to move decision support off machine
	Stored procedures to reduce compilation time and network usage
	Minimum locking level that meets application needs
Database	Transaction log thresholds to automate dumps and avoid running out of space
	Thresholds for space monitoring in data segments
	Partitions to speed loading of data
	Devices to avoid disk contention, take advantage of I/O parallelism
Server	Tuning memory, most critical configuration parameters and other parameters
	Configuring cache sized and I/O sizes
	Scheduling batch jobs and reporting for off hours
	Reconfiguring parameters for shifting workload patterns
Devices	More medium-sized devices and more controllers for better I/O throughput
	Distributing databases, tables, and indexes for even I/O load across devices
	Segments, partitions for I/O performance on large tables used for parallel queries
Network	Configuring packet sizes to match application needs
	Configuring subnets
	Isolating heavy network uses
	Configuring for multiple network engines
Hardware	Configuring the housekeeper task to improve CPU use
	Configuring multiple data caches
Operating System	Choosing between riles and raw partitions
	Increasing memory size

This page intentionally left blank.

Ensuring Database Quality

Integrity Monitoring

The Database Consistency Checker (**dbcc**) is a set of utility commands for checking the logical and physical consistency of a database. Use the **dbcc** commands:

- As part of regular database maintenance (periodic checks run by the System Administrator). These checks can detect, and often correct, errors before they affect a user's ability to use SQL Server.
- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database.
- Because you suspect that a database is damaged. For example, if using a particular table generates the message "Table corrupt", you can use dbcc to determine if other tables in the database are also damaged.

The integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency checks on a regular basis. Two major functions of dbcc are:

- Checking allocation structures (the commands checkalloc, tablealloc, and indexalloc).
- Checking page linkage and data pointers at both the page level and row level (checktable and checkdb). The next section explains page and object allocation and page linkage.

dbcc is used in the database backup scripts to determine if the database is properly configured and without errors. If errors occur, the backup will not proceed and a message is sent to the DBA with notification of the problem.

This page intentionally left blank.

Sybase Troubleshooting

Space Usage

Thresholds are defined on segments to provide a free space value at which a procedure is executed to provide a warning or to take remedial action. Use **sp_addthreshold** to define your own thresholds**: sp_addthreshold database_name, segment_name, free_space, procedure_name**, where **free_space** is the number of free pages at which the threshold procedure executes; **procedure_name** is the stored procedure which the threshold manager executes when the number of free pages falls below the **free_space** value.

Deadlocks

A deadlock (also known as a "deadly embrace") is a situation where two database processes are simultaneously attempting to lock data that the other holds. For example, two users (A and B) are updating the same table of data at the same time. User A holds a lock on Page 1 and requests a lock on Page 2. Meanwhile, user B holds a lock on Page 2 and has requested a lock on Page 1. Without intervention, these two jobs would never finish.

Sybase detects these situations, analyzes the two processes and automatically kills the process with less accumulated processor time. Sybase prints out an error message to the killed process and requests that the user re-submit the job. A Sybase error message similar to the following one is displayed:

> **Msg 1205, Level 13, State 1:**
> **Server 'SERVER', Procedure 'sp_whatever', Line 123:**
> **Your server command (family id #0, process id #99) was deadlocked with another process and has been chosen as deadlock victim. Re-run your command. (return status = -3)**

Troubleshoot Chronic Deadlock

1 Turn on deadlock printing information and **sp_configure** "print deadlock information".

2 Recreate the problem.

3 As deadlocks occur, **tail -f the $SYBASE/install/errorlog** (or wherever the errorlog prints out to).

- Detailed information is printed out (e.g., line numbers of code causing deadlocks, tables within Sybase).

4 Run dbcc traceon(3604) and dbcc page(#) on the page numbers printed by traceflag 1204.

- dbcc page will report the tablename and the index that is being used.

- It is helpful to see if it is a non-clustered or a clustered index.

5 Monitor locks and blocked processes during deadlock re-creation.

- Use **sp_lock**, **sp_block** and **sp_blocker** to print out detailed information about locking and blocking.

6 Run **sp_sysmon** while re-creating the problem, and analyze the "Lock Management" section.

- Advanced deadlock diagnosis can be conducted using:
  - **traceflags 602, 603**, which prints out information for deadlock prevention.
  - **traceflag 1205**, which puts a stack trace on the deadlock.
  - **traceflag 8203**, which displays statement and transaction locks.
  - **sp_configure** "deadlock checking period", "deadlock retries", which are configuration parameters related to deadlocking.
- Common deadlock problems and solutions include:
  - **Contention on heap tables**. "Last page contention" problems occur where high insert rates always seem to go on the last page of a table. This creates what is known as a "heap table," which is bound to have performance problems. The end of the table (in terms of a page chain) is a hotspot of activity. If this is the cause of the deadlocks, either partition the table (which creates several insert points for data and eliminates the "heapness") or re-create your clustered index on a field that spreads inserts along the entire page chain of the table (i.e., clustered indexes on surrogate keys as opposed to last name or something similarly random). This can be seen in **sp_sysmon** output "Last page locks on heaps."
  - **Poorly written key generation schemes**. Using a "select max(col)+1" method or high rate OLTP applications generating keys from a badly configured **key_storage** table system are always deadlock candidates. Never use the "select max+1" method. If you must use a **key_storage** solution (if you can't deal with identity gaps for example), ensure that the key_storage table is configured for Row Level Locking (in Sybase 11.9.2 and above) or has max_rows_per_page=1 (in Sybase 11.5 and 11.0).
  - **Lock contention; sp_configure "number of locks"**. If you have configured too few locks for the system, severe lock contention can occur and deadlocking can be common. There is an entire section **of sp_sysmon** devoted to Lock management. Large updating within multi-user applications and long running transactions will definitely lead to deadlocking. Don't issue multiple update statements within the same transaction. If you must update multiple objects within the same transaction, make sure you consistently access the objects in the same order throughout the application. Updating or deleting from tables

without covering indexes cause table scans. Other options: System data Lock strategies: In 11.9.2 and above: examine whether it may be wise to go to Row Level locking or Data only locking.

- **Hardware problems**. Slower I/O devices increase deadlock vulnerability.
- **Cursors**. Should be avoided.
- **Parallelism**. y using some of the parallel processing features available in later versions, deadlocking is subdued.
- **ANSI isolation level 3 (**serialization) locking. Avoid when possible.
- **Descending scans in indexes**. Can sometimes cause deadlocks after upgrading where none previously occurred.
- **Application-side deadlocking**. See Section 1.5.1 of the Sybase FAQ for a quick example of Application-side deadlocking and how to avoid it.
- **Deadlocking increases with upgrade**. These deadlocks are frequently caused by the "allow backward scans" option being turned on by default. They can also be caused by differing optimizer paths with new version (after upgrade) or just the engine's sheer speed. Also recommended to increase lock escalation threshold if using RLL (defaults to 200, which is not a lot of rows).

This page intentionally left blank.

Oracle Procedures

Oracle Operating System Environment

The following items identify the location of oracle components:

- ORACLE_HOME : Specifies the directory where the Oracle software will be installed /usr/ecs/OPS/COTS/oracle/8.1.6
- ORACLE_SID : Specifies the instance name. SID : pds, pdsit, pdsst.
- ORACLE_BASE : /usr/ecs/OPS/COTS/oracle

Starting Up the Database

To start the Oracle database, use the following procedure.

Starting the Oracle Database

1 Log in to the appropriate host.

2 To start up the instance and open the database, type **STARTUP** and then press **Return**.

3 To start up the instance only, type **STARTUP NOMOUNT** and then press **Return.**

4 To change the state of the database from NOMOUNT to MOUNT, type **ALTER DATABASE MOUNT;** and then press **Return**.

5 To open the database, type **ALTER DATABASE OPEN;** then press **Return**.

Shutting Down the Database

To shutdown the database and close the instance, the following procedure is applicable.

Shutdown the Oracle Database

1 On the appropriate host, type **SHUTDOWN IMMEDIATE** and then press **Return**.

2 Type **SHUTDOWN** and then press **Return**.

- This is the normal shutdown. Oracle waits for all users to disconnect before completing the shutdown.

Controlling the Listener

The listener is the process on the server that listens to the connection request from the client's machine, e.g., pdssa. The following commands control the listener:

- Lsnrctl start = startup a listener.
- Lsnrctl stop = shutdown a listener.

Data Dictionary View Categories

The following commands specify views:

- DBA_xxx : Database Administrator's view.
- ALL_xxx : Expanded user's view.
- USER_xxx : User's view.

Example: Run a select statement for the Database Administrator's view of the following database information.

- General overview : DICTIONARY
- Schema objects view : DBA_OBJECTS, DBA_TABLES, DBA_TAB_COLUMNS, DBA_CONSTRAINTS
- Space allocation view : DBA_SEGMENT, DBA_FREE_SPACE, DBA_ENTENTS
- Database Structure : DBA_DATA_FILES, DBA_ROLLBACK_SEGS, DBA_TABLESPACES

Obtaining Information and Controlling the System

There are many commands for managing the Oracle database. Table 17 lists a few that may be particularly useful.

Table 17. Some Useful Oracle Commands

Command	Function
ARCHIVE LOG LIST;	Displays the current ARCHIVELOG status of the database from within the Server Manager (the system must be in ARCHIVELOG mode to enable online or “hot” backups)
SELECT * FROM V$DATABASE;	SQL SELECT displays information from specified database table
SELECT GROUPS,CURRENT_GROUP#,SEQUENCE# FROM V$THREAD;	Displays information about log file groups
SELECT GROUP#MEMBERS,STATUS FROM V$LOG;	Display information about log file groups and members
ALTER SYSTEM SWITCH LOGFILE;	Forces Oracle to begin writing to a new redo log file group, regardless of whether the files in the current group are full
ALTER SYSTEM CHECKPOINT;	Forces Oracle to perform a checkpoint (i.e., ensures that all changes made by committed transactions are written to the data files on disk

Oracle Troubleshooting

The following control file parameters need to be examined:

- BACKGROUND_DUMP_DEST : location where background process trace files are written. This is also the location for the alert log.
- DB_BLOCK_BUFFERS : Number of blocks cached in the SGA.
- SHARED_POOL_SIZE : Size in bytes of the shared pool.
- USER_DUMP_DEST : Location where user debugging trace files are created o behalf of a user process.
- PROCESS : Maximum number of operating system processes that can connect simultaneously to this instance.

Check each database’s error log at the appropriate location:

- /usr/ecs/OPS/COTS/oracle/admin/pds/
- /usr/ecs/OPS/COTS/oracle/admin/pdsst/
- /usr/ecs/OPS/COTS/oracle/admin/pdsit/

Accessing Dynamic Performance View

Dynamic performance view can be accessed in the NOMOUNT stage. Run a select statement to view the dynamic database server performance.

- Select * from V$PARAMETER
- Select * from V$SGA
- Select * from V$PROCESS
- Select * from V$SESSION
- Select * from V$INSTANCE
- Select * from V$CONTROLFILE
- Select * from V$DATABASE
- Select * from V$DATAFILE
- Select * from V$LOGFILE
- Select * from V$TABLESPACE
- Select * from V$LOG

Displaying Parameter Values

To display a parameter value, type **SHOW PARAMETER;** then press **Return**.

Displaying Information about Users

To display information about users, type **SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS;** then press **Return**.

Displaying information about system privileges and object privileges

The following commands (followed by a press of the **Return** key) are useful to obtain information about system privileges and object privileges:

- SELECT * FROM DBA_SYS_PRIVS;
- SELECT * FROM SESSION_PRIVS;
- SELECT * FROM DBA_TAB_PRIVS;
- SELECT * FROM DBA_COL_PRIVS;

Terminating Sessions

The following procedure is used to kill a user process or abnormal session.

Terminating a Session

---

1 **SELECT SID, SERIAL# FROM V$SESSION WHERE USERNAME=”*xxxxxxx*”;**

2 Write down the sid and serial# from above query.

3 Run **ALTER SYSTEM KILL SESSION ‘*integer1*, *integer2* ;**

- ***integer1***: value of the SID column.
- ***integer2***: value of the SERIAL#.
- When the ACTIVE user session is terminated, the transaction is rolled back and the user immediately receives the following message.

  ORA – 00028 : your session has been killed.

This page intentionally left blank.

Practical Exercises

Introduction

This exercise is designed to practice key elements of the database administration procedures. Perform the tasks identified in the exercise.

Equipment and Materials

One ECS workstation, a copy of 609-EMD-001, *Release 7 Operations Tools Manual for the EMD Project*, and a copy of 611-EMD-001, *Mission Operation Procedures for the EMD Project*.

Starting and Stopping SQL Server

Check to see that the SQL Server processes are running.

- If they are not running, start them.
- If they are running, shut them down and restart them.

Creating Database Devices

1 Create the script file(s) that will create a database device on a server.

- The following parameters should be met:

  Device Name = class_device

  Physical Name = TBD

  Device Size = 100 MB (be sure to convert this to blocks!)

  Virtual Device Number = number you have determined.

- Be sure to prepare all the appropriate files, properly named and located in the correct subdirectories.

2 If time and space permit (to be determined by the instructor), perform the creation of this device.

Configuring Databases

1 Start the Configuration Registry GUI and display configuration parameters for a database server of your choice.

2 Practice the procedure for changing a configuration parameter up to, but not including, confirmation of the change.

Monitoring Database Performance

Start **sp_sysmon** and observe cache usage and network traffic on the system.

Backing Up Databases

1 Perform a manual backup of the AutoSys database. Send the backup to the **autosys_backup** server.

- Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.

2 Assume that the database device you created at the beginning of these exercises has failed. Perform all the steps required to recreate the database device and restore the database to full functionality.

Case Study: Sybase Troubleshooting

Assume there is a deadlock on the Storage Management database. Execute the procedure for troubleshooting chronic deadlocks on the database host.

Slide Presentation

Slide Presentation Description

The following slide presentation represents the slides used by the instructor during the conduct of this lesson.

This page intentionally left blank.